



MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-2007-001

Using the Inverted Classroom to Teach Software Engineering

Gerald C. Gannod , Janet E. Burge, Michael T. Helmick



Using the Inverted Classroom to Teach Software Engineering

Gerald C. Gannod^{*}
Department of Computer
Science and Systems Analysis
Miami University
Oxford, OH 45056
gannodg@muohio.edu

Janet E. Burge
Department of Computer
Science and Systems Analysis
Miami University
Oxford, OH 45056
burgeje@muohio.edu

Michael T. Helmick
Department of Computer
Science and Systems Analysis
Miami University
Oxford, OH 45056
mike.helmick@muohio.edu

ABSTRACT

An inverted classroom is a teaching environment that mixes the use of technology with hands-on activities. In an inverted classroom, typical in-class lecture time is replaced with laboratory and in-class activities. Outside class time, lectures are delivered over some other medium such as video on-demand. In a three credit hour course for instance, contact hours are spent having students actively engaged in learning activities. Outside of class, students are focused on viewing 3-6 hours of lectures per week. Additional time outside of class is spent completing learning activities. In this paper we present the inverted classroom model in the context of a software engineering curriculum. The paper motivates the use of the inverted classroom and suggests how different courses from the Software Engineering 2004 Model Curriculum Volume can incorporate the use of the inverted classroom. In addition, we present the results of a pilot course that utilized the inverted classroom model at Miami University and describe courses that are currently in process of piloting its use.

Categories and Subject Descriptors

D.2.0 [Software Engineering]: General; K.3.2 [Computing Milieux]: Computers and Education—Curriculum, Computer science education

General Terms

Software Engineering Education

Keywords

Inverted Classroom, Technology in Education, Podcasting

^{*}Contact Author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICSE '08 Leipzig, Germany

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

1. INTRODUCTION

Software engineering is, at its essence, an applied discipline that involves interaction with customers, collaboration with globally distributed developers, and hands-on production of software artifacts. The education of future software engineers is, by necessity, an endeavor that requires students to be active learners. That is, students must gain experience, not in isolation, but in the presence of other learners and under the mentorship of instructors and practitioners.

An *inverted classroom* is a teaching environment that mixes the use of technology with hands-on activities [1]. In an inverted classroom, typical in-class lecture time is replaced with laboratory and in-class activities. Outside class time, lectures are delivered over some other medium such as video on-demand. In a three credit hour course for instance, contact hours are spent having students actively engaged in learning activities. Outside of class, students are focused on viewing 3-6 hours of lectures per week. Additional time outside of class is spent completing in-class activities.

Past uses of the inverted classroom have included the use of video tape, DVD players and downloadable media files [1]. New technologies, such as iPods, and new broadcast methodologies, such as *podcasting*, have made access to multimedia data more accessible and ubiquitous. While many educators are exploring a wide variety ways to utilize the iPod for instruction [2, 3], there has yet to be any consensus on the most effective use of the technology in the classroom.

In the inverted classroom, learning activities that would typically are done outside of class, are done in-class in the presence of the instructor. Passive activities, such as listening to lectures, are performed outside of class. As a result, valuable faculty “face” time is not spent merely communicating information, but rather is spent engaging directly with students when they are involved in in-depth learning activities.

Many computing courses ultimately involve hands-on activities. While significant long-term projects are a staple of computing programs, short high-payoff homework and lab assignments can benefit greatly from a more traditional laboratory environment. In this paper we present the inverted classroom model [1] in the context of a software engineering curriculum. The paper motivates the use of the inverted classroom and suggests how different courses from the Software Engineering 2004 Model Curriculum Volume [4] can incorporate the use of the inverted classroom. In addition, we present the results a pilot course that utilized the inverted classroom model at Miami University and describe

courses that are currently in process of piloting its use.

2. BACKGROUND

This section presents the context of the work described in this paper.

2.1 Educational Models

Cooperative or collaborative learning was derived from theories relating to motivation and movement towards desired goals. The idea of cooperative learning comes from the idea that a social group (e.g., students) when linked together can achieve some positive goal (e.g., learning) [5]. Collaborative learning has become popular recently as a means for providing instruction, especially in a group context [6]. The origins of *distance education* harken back to days of correspondence via mail and other asynchronous media as a means for providing instruction. Distance education has become more popular with the inception of the Internet. Institutions such as the University of Phoenix and the Open University have fully embraced the use of distance education. In this paper we describe an approach to learning that utilizes the benefits of both collaborative and distance learning.

2.2 Software Engineering Curricula

Software Engineering Curriculum 2004 Volume.

The 2004 Software Engineering Model Curriculum was produced as a resource for institutions wanting to propose software engineering degree programs [4]. The volume provides a detailed set of models that are tailorable to different organization types and includes model curricula that are applicable for introducing software engineering in either the 1st or 2nd year of a program. Different models have also been produced to account for location of an institution (e.g., Europe vs. North America vs. Australia, etc.).

Software Factory.

Several models for delivering software engineering content have been suggested, ranging from capstone courses to industry-required projects. One such model has been suggested by Tvedt, et al. on the *software factory* [7]. Their model introduces the notion of using cohorts in a staggered manner. In the model, students are involved in developing software as part of project teams with advanced cohorts leading projects and beginning cohorts acting as development teams. This particular model is interesting from the standpoint of the “Montessori” type model of instruction as well as in the amount of hands-on activity that occurs on the projects. In this paper we describe how an inverted classroom model increases hands-on experience but look at individual courses rather than a reordering of an entire curriculum.

2.3 Podcasting for Education

A “podcast” is a term used to describe the use of a subscription-based broadcasting of video and audio content using *really simple syndication*. The original intent was for the use of podcasting to push content to owners of Apple iPods, although podcasts are not limited to use of by just iPod owners. The use of podcasting for education has seen increased adoption as evidenced by the amount of content now available at the Apple iTunesU site [3]. However, the focus of

podcasting in education has been on production and technology [2], and little on the pedagogy of using podcasting. In this paper, we present a model of education that is facilitated by podcasting, namely the *inverted classroom*.

3. INVERTED CLASSROOM

3.1 The Learners

The student of today, the so-called “Millennial”, works from a mindset that is different than each of the preceding “Gen-Xer” and “Boomer” generations. Frand identifies ten characteristics common with millennials [8]:

1. *Computers aren't technology* - millennial students have grown up in an environment where computers and the Internet are ubiquitous. Computers are not a new technology.
2. *The Internet is better than TV* - the number of hours spent on the Internet has increased while the amount of time watching television has decreased.
3. *Reality is no longer real* - images and other things viewed on the Internet or on TV may have been altered. There is little trust for authenticity of many things.
4. *Doing is more important than knowing* - the activity of accumulating knowledge is viewed as less important than gaining skills that enable them to deal with complex and ambiguous information.
5. *Learning more closely resembles Nintendo than logic* - the trial-and-error mentality that comes from the experience millennials have from playing video games is far more pervasive than in previous generations.
6. *Multitasking is a way of life* - it is not uncommon to find young people doing many things at once (e.g., simultaneously listening to music, eating, sending instant messages and watching TV)
7. *Typing is preferred to handwriting* - millennial students prefer using word processors or other computing based recording mechanism over writing.
8. *Staying connected is essential* - millennial students are continuously connected using a plethora of devices including cell phones, computers, and other hand-held devices.
9. *There is zero tolerance for delays* - while some of us can remember the days when TV stations would go off the air, millennial students expect 24x7 access to services and people.
10. *Consumer and creator are blurring* - there is a belief that there is little difference between the owner, creator, and user of information.

Foreman identifies a number of learning theory essentials [9]. Specifically, he states that the ideal learning situation *is customized, provides immediate feedback, is constructive, motivates students to persist, and builds enduring conceptual structures*. In regards to customization, Foreman indicates that optimal learning addresses learning styles and “proximal zones”. That is, optimal learning should not appear

foreign to a learner. Optimal learning is constructive means that allows students to explore multisensorial environments through active discovery. Persistence refers to motivating students to gain a desire to pursue more knowledge in a particular area. Finally, optimal learning promotes development and commitment of knowledge to long-term memory in order to integrate that knowledge for everyday practical usage.

As educators of software engineering, it is our task to determine how we can connect millennial students that are described as Frand indicates, with the characteristics of the ideal learning environment described by Foreman.

3.2 Inverted Classroom

The traditional *instructor-centered* educational model is based on the use of class contact time (e.g., class time) on the delivery of information through lecture. In this model, the expectation is that an instructor impart knowledge of some particular topic. A traditional model is limited by the constraint that any particular class meeting is limited to a finite number of minutes, and that there are a finite number of class meetings. The challenge often associated with instructor-centered education is the lack of in-class active learning. Specifically, while instructors often do mix lecture with in-class activities that facilitate active learning, there is a tension between use of that class time for those activities versus the need to “cover” topics found on a syllabus.

More progressive models of instruction includes *collaborative learning*, where students are focused on some particular task and must, as a group, identify the relevant topics, theories, and methodologies that are relevant in completing that task. For instance, the work by Dietrich and Urban demonstrated the use of collaborative and active learning in database courses [6]. A potential problem with collaborative learning lies in the fact that it becomes difficult to assess whether certain educational outcomes are achieved. Thus, collaborative learning must be tempered with an appropriate amount of instructor-centered lecture to ensure topic coverage.

A growing trend towards *distance learning*, especially in for-profit institutions, has taken advantage of the Internet by providing access to content for use by students in self-paced environment. The benefit of distance learning is that the learner can access information at their own pace and can continually reference recorded material. That is, in some models where lecture materials are provided through a recorded medium, students can pause, fast-forward, reverse, and replay lectured content. Such models rely upon student motivation to manage course requirements and learning activities and thus self-motivation is paramount for consistent success. Finally, much of the learning that occurs in such environments is asynchronous and thus the ability to provide the benefits of collaborative and active learning is non-existent.

An *inverted classroom* approach for instruction “inverts” the traditional instructor-centered model while taking advantage of the benefits of both distance learning and collaborative and active learning [1]. In an inverted classroom, lecture content is provided over some asynchronous medium. Students access the lecture content outside of class during the non-contact hours of a semester. During the contact hours (e.g., the “normal” class periods) students are involved in learning activities in the form of in-class assignments, lab-

oratories, and discussions. Table 1 shows some of the differences between the inverted classroom and the traditional lecture model.

The benefits of using an inverted classroom model are many. First, with respect to *coverage*, since the lecture content is delivered asynchronously, there are no limitations imposed by a finite number of class minutes or meetings. Second, again since the content is delivered asynchronously, students can access, view, and review material at their own pace. Third, by having the in-class component of the course, students can be engaged in active learning with other learners on a regular basis. Specifically, by having the students working in a laboratory environment where discussion between peers is encouraged, learners can take advantage of the benefits that result from having to explain concepts to each other, thus reinforcing and solidifying their own understanding of those concepts. From the standpoint of active versus passive learning, the inverted classroom model has the effect of pushing passive learning (e.g., listening to lectures) away from the classroom and to the home, library, or other viewing location. Active learning, then, is pulled to the classroom. In addition, since the instructor is freed up from having to lecture during the in-class period, the instructor is able to engage with the students when the active learning is occurring.

3.3 Learners and the Inverted Classroom

The inverted classroom model addresses many of the Frand characteristics of the millennial student. Specifically, the inverted classroom addresses the following characteristics:

4. *Doing is more important than knowing* - the inverted classroom model takes the focus away from the lecture and places it upon the extensive use of active learning. Software engineering is a highly applied activity. Learning in this context depends on repeated application of techniques in order to gain experience.
5. *Learning more closely resembles Nintendo than logic* - the inverted classroom provides more opportunity for iteration. Software engineering processes are highly iterative. The ability to define and refine different artifacts in the collaborative environment provided by the inverted classroom offers many opportunities to use iteration to refine artifacts and thus reinforce student knowledge and capabilities.
6. *Multitasking is a way of life* - content delivery through podcasted lectures allows students to do something that comes natural to them. Since content is delivered via podcasting, students can take advantage of the ability to pause, restart, and review lectures at their leisure or in and amongst many of the tasks they may be undergoing at any given time.
9. *There is zero tolerance for delays* - the inverted classroom facilitates providing immediate feedback when that feedback is most important. In the inverted classroom model, feedback can be provided immediately during the in-class contact hour when learning activities are being performed. In addition, if an instructor is brave enough to wade into the instant messaging waters, immediate feedback can also be given at other times.

	Instructor		Student	
	Traditional	Inverted	Traditional	Inverted
Prep for lecture	Standard prep time	Standard prep plus prep for additional material (as desired), produce recorded lecture (1st offering) at least two days prior to contact hour	N/A	N/A
Prep for class	Same as lecture prep	Develop learning activity before meeting time	Readings	Readings, view podcasts before class
Attendance	N/A	N/A	Only if required	Required
Learning Activities	Instructor feedback delayed, contact and guidance limited to office hours	Instructor feedback in process, contact during entire contact hour	Outside class	In-class / Outside class

Table 1: Differences between Traditional and Inverted Classroom Models

With respect to Foreman’s optimal learning situations, the inverted classroom is:

- *Customized* - the use of podcasting allows the student to focus on passive content as much as needed, when needed. The hands-on learning activities facilitate customized instruction by allowing the instructor to be more involved in the active learning of the students.
- *Provides Immediate Feedback* - the in-class activities during contact hours allow the instructor to provide immediate feedback on a more regular basis.
- *Constructive* - the combination of the use of lecture, screencasts video blogs, and other supplemental video along with a more hands-on classroom experience, exposes students to a constructive environment rife with active discovery.
- *Motivating* - persistent hands-on activities allow students to observe the purpose for many of learning outcomes for a course.
- *Enduring* - the inverted classroom provides an opportunity for reinforcement of concepts. The reinforcement and hand-on application assists in helping students commit knowledge to long-term memory.

3.4 Instructors and the Inverted Classroom

Unfortunately, the success of a new teaching paradigm does not rest solely on its ability to affect student learning. The most successful educational initiatives are those that provide benefits to both teacher and learner. The inverted classroom falls into that category in a number of ways.

First, it puts the primary focus of the class on the part of teaching that most professors find the most rewarding: interaction with their students. Even the most interactive lectures are likely to actively involve only a subset of the students. In the inverted classroom, the instructor works directly with individual students during contact hours. Most

of their time, in this model, can be spent with those students that are struggling, as opposed to the traditional lecture where most of the questions posed during discussion come from the stronger students (while the strugglers are more likely to either be absent or sitting quietly at the back of the room).

Second, in-class hands on activities not only engage the learner they engage the instructor. The conventional wisdom in teaching is that the best class taught on a subject is when an instructor is teaching it for the third time. During the first time, the primary goal is instructor survival. During the second, much of the preparation time is spent fixing the initial mistakes. Then, on the third teaching, the instructor has confidence with the material and their presentation. So what about the next few times? The risk then is that boredom sets in. If the instructor is not excited about the material it is hard to hide it from the students. Also, even the most interesting class has at least one topic that the instructor dreads teaching. For a computer architecture course, it may be binary arithmetic. For software engineering, it may be configuration management. With the inverted classroom, an instructor gives the lecture once, adding changes as needed from semester to semester, and instead can focus the bulk of their time and energy on the part of the class that is exciting and different from semester to semester: their students. Each semester brings a fresh group of students with their own individual approaches to the material: the best antidote to instructor boredom.

Third, the inverted classroom provides an easy way to involve guest speakers in classroom instruction. In a class that covers a broad subject area, such as software engineering, not all instructors will be equally adept at all topics. In addition, it may be beneficial to bring in some “outside voices” such as bringing in experienced industry professionals. Trying to schedule guest speakers for a traditional lecture can be difficult. With the inverted classroom, speakers can deliver their portion of the lecture at their convenience. This relieves the instructor from having to structure their

syllabus around guest speaker availability. Podcasting guest speakers also removes the risk of a guest speaker turning what should be an instructional experience for the students into a recruiting pitch for their company.

3.5 Lectures through Podcasting

In previous work, we described how we used *podcasting* as the preferred medium for delivering content to students within an inverted classroom setting [10]. In that work, course lectures were produced as podcasts approximately one week prior to the corresponding assignments. The software used to produce the podcasts (all for the Mac) included ProfCast [11], for capturing Microsoft Powerpoint and Apple Keynote presentations with voice overs, Snapz [12], for capturing full-motion presentations of software use (e.g., a “screencast”), iMovie [13], for capturing full-motion talking head lectures, and iWeb [13], for deploying the podcast on a standard web server. Blackboard [14] was used to save and deploy Powerpoint and PDF files, as well as for gradebook and assignment management. Students used either the iTunes music software system as a podcasting client, or a non-podcast client such as a web browser to view video on a web page.

For our current semester’s pilot courses using inverted classroom techniques, all content is being delivered through the Computer Science Courseware System (CSCW) [15]. The CSCW system has support for delivering podcasts directly to podcasting client programs, such as Apple’s iTunes. Students are free to receive lecture podcasts automatically or to interactively download the content direct from the CSCW Web site.

3.6 Learning Activities

Table 2 shows a general comparison of the quality and depth of learning activities for traditional and inverted classroom models of instruction. The experience of instructors may vary, but in general these comparisons hold. The number of assignments, when using the inverted classroom model, can be much higher than in the traditional classroom model. Specifically, for some courses there can be one assignment per course contact hour (minus exams). As a result, it is much easier to have learning activities address specific outcomes. This is contrasted with traditional homework assignments or projects that might target several learning outcomes at once. The feedback that can be provided to a student in the inverted classroom model is in many instances immediate. The level of interaction with students during course contact hours provides the ability to point students in the right direction and to give guidance as needed. In this way, students can be steered away from pitfalls and incorrect assumptions, allowing the students to use trial-and-error in their problem solving process.

Finally, with respect to assignment depth, learning activities in the inverted classroom model, due to time constraints, contain less depth than in the traditional model. This can be alleviated by creating assignments that span longer periods of time (two class periods). One model that has been employed is to assign multi-part assignments with initial completion occurring in class, and the final completion occurring outside of class.

3.7 Issues

A number of concerns exist regarding the inverted class-

Inverted		
Traditional		
Number of Assignments	Low	High
Outcome coverage	Low	High
Feedback	Delayed	Immediate
Depth per assignment	High	Low

Table 2: Comparison of Learning Activities

room model of instruction. In this section, we describe each and provide discussion.

I like to interact with students during lecture.

Many times during lecture in the traditional lecture model, questions and discussion cause a “light” turn on in a student’s face. The lack of interaction in a podcasted lecture, as such, removes the ability of a student to ask a question to clarify some idea. To address this concern, students are asked to write down their questions and the time index of the podcasted material and to bring those questions to class. A certain amount of time is then devoted at the beginning of class to answer those questions. In addition, by using instant messaging, students can ask questions regarding a lecture being viewed. The instructor has the option of declining the message or answering it.

Interaction with students is a cornerstone of the inverted classroom experience. The amount of interaction with students actually increases with this model. Anecdotaly, we have found that the increased contact has made it easier to identify which students are struggling and which students are excelling.

Do students come to class?.

Student attendance should be mandatory when using the inverted classroom model. Since learning activities equate to graded homework assignments, there is often no choice as to whether students must attend or not. Skipping a class period results in a loss of points.

Do students watch the podcasts?.

It must be stressed to students that in order for students to be able to complete in-class assignments, the podcasts must be viewed and notes taken. By composing the learning activities so that it relies on the lectured material, students quickly learn that the lecture materials are important and must be viewed. Our experience has shown that failure to keep current on the lectured material often results in inability to complete assignments.

What is the overhead?.

The startup overhead of adopting the inverted classroom model can be significant. In the first semester that a course is taught using this model, lectures must be produced. From the standpoint of preparation time, the difference between the inverted classroom model and the traditional model is little. However, the production of the podcasts amounts to setting aside the time to record the lectures, which ends up being added up as prep time. Normally, the lecture would just be delivered during the class contact hours, which for a

3 credit hour course amounts to about 3 hours a week. The amount of time “lecturing” in the inverted classroom model varies since the amount of content to be delivered can be as little or as much as is wanted. Fortunately, in subsequent semesters, the amount of time producing lectures becomes much smaller since only short addendums or updates are needed.

Another aspect of the overhead is the production of learning activities. For some courses, new activities will need to be developed each semester. The number of learning activities that are produced increases significantly with the inverted classroom model since you may need to have one assignment ready per class period. Grading also increases since learning activities need to be evaluated. However, you can incorporate learner peer evaluation more readily in the inverted classroom model.

It doesn't fit my style of teaching.

The inverted classroom model is learner-centered. It focuses primarily upon the student and upon increasing the amount of interaction between the student and instructor. The approach is intended to address issues related to the millennial student and less upon the instructor teaching the millennial. Ultimately, as with any teaching method, the most important factor is the learning outcome. Instructors may be effective in any style of teaching; the inverted classroom model is an alternative.

How does this work for large classes?.

The success of the inverted classroom model is dependent on the ability of the instructor to interact with the students as they complete their in-class activities. At Miami University, the class sizes for the pilot courses was twenty-four (24) in Spring 2007. In our current semester's pilot courses, the enrollment is forty-three (43) students for two sections of a data structures course and over eighty (80) students for three sections of a programming fundamentals course. For larger classes, providing the desirable amount of instructor-student interaction would necessitate breaking these classes into smaller sections and/or providing support from teaching assistants. For most software engineering courses, this model would require computer equipped classroom laboratories so that the students would have access to the hardware and software required to complete their projects. Many schools with large Computer Science programs already follow a lecture-lab model where lectures are taught by instructors and labs by teaching assistants. For the inverted classroom, it is important that the instructor themselves be present and involved during the in-class activities in order to realize many of the benefits of this approach.

4. SE CURRICULUM

Software engineering is a process-centric discipline. The education of students in this field is best achieved through repetitive, hands-on, activities and projects in a collaborative environment that fosters communication between stakeholders. By using the inverted classroom, these activities can be easily modeled and repeated in order to provide students with a strong foundation on various aspects of the field.

In this section, we present a model for integrating the use of the inverted classroom model for different courses

into the software engineering curriculum. The curriculum that we present is based on the IEEE/ACM Software Engineering Model Curriculum [4] (e.g., the SE 2004 Volume) and focuses primarily upon the software engineering specific courses. The entire curriculum includes some overlap with the Computer Science Model Curriculum [16]. In the discussion below, we consider the Computer Science courses that serve as a direct line of pre-requisites for the SE 201 Introduction to Software Engineering course. Finally, the model focuses on starting software engineering in the second year [4].

4.1 Overview of SE Courses

The core software engineering curriculum is composed of the following sequence of courses [4]:

- CS 101I Fundamentals of Programming** - covers fundamental topics in programming including control structures. The “I” indicates an “imperative first” treatment of computing.
- CS 103I Data Structures and Algorithms** - covers data structures and data abstractions.
- SE 201 Introduction to Software Engineering** - covers the foundations for software engineering by covering the principles and concepts of the field.
- SE 211 Software Construction** - covers low-level design issues
- SE 212 Software Engineering Approach to HCI** - covers the design and implementation of user interfaces
- SE 311 Software Design and Architecture** - covers advanced software design including distributed systems and software architecture
- SE 321 Software Quality Assurance and Testing** - covers software quality and testing
- SE 322 Software Requirements Analysis** - covers software requirements elicitation, specification, and analysis
- SE 323 Software Project Management** - covers project management issues
- SE 400 Software Engineering Capstone** - provides students experience in working on a year long project
- SE 4xx Software Engineering Special Topics** - special topics in software engineering; content determined by faculty

4.2 Inversion

In the remainder of this section, a model of how inversion applies to each of the courses listed above is presented. The content for each of the courses, as delivered through podcasting or some other medium, remains the same as is defined in the SE 2004 Volume [4]. Below we describe how learning activities can be structured during course contact hours to take advantage of the inverted classroom model.

CS 101I Fundamentals of Programming.

The Computing Curricula 2001 Volume does not give specific guidance on learning activities for this course [16]. However, this course is programming intensive, and thus is naturally suited for a laboratory dominated experience. The course contact hours in this course can be devoted to programming assignments performed in class.

CS 103I Data Structures and Algorithms.

As with the CS 101I course, the data structures and algorithms course is programming intensive and thus the course contact hours can be dominated by programming activities.

SE 201 Introduction to Software Engineering.

The traditional model for teaching SE 201 involves presenting lectures on a wide variety of topics ranging from software project inception through delivery and maintenance. In our experience teaching this course using a traditional lecture model, the homework assignments are used to provide learning activities on each of the major topics, and a significant semester long project provides teaming experience. Due to the amount of content in the course, in-class activities in the form of laboratories or small group experiences are limited. The use of inversion in this course alleviates three problems: coverage, experience, team coordination. With respect to coverage, use of the inverted classroom model allows the instructor to cover as much material as desired in the podcasting (or other delivery) format. In regards to experience, the freed contact hours can be used by the instructor to model each of the software engineering activities in detail. For instance, when performing the requirements elicitation activity, the instructor can use the time to discuss an example of eliciting requirements, and involve the students in that example. In regards to team coordination, the contact hours can be used by student groups to hold meetings. The instructor can then be present during a number of those meetings to provide guidance, answer questions, or to observe student decision making processes.

SE 211 Software Construction.

The focus of the Software Construction course in the SE 2004 curriculum is on low-level design issues such as the use of parsers, the definition of protocols, application of formal methods, and tools for a wide variety of purposes including debugging, performance tuning, and model-driven development [4]. With such a wide variety of topics, all of which are tool driven and require knowledge and experience in the use of those tools, the application of the inverted classroom is natural. The course contact hours can be devoted to short in-class assignments that focus on the use of the tools. The benefit of the inversion of this course is that the instructor and teaching assistants can be present in laboratory-like environment, providing guidance on how to resolve various issues that arise in using the tools. In our experience with similar kinds of environments, we have found that short single session assignments provide students with initial experience on getting started with a technology. Longer multi-session assignments, on the other hand, provide students the opportunity for self-paced discovery needed for optimal learning to occur.

SE 212 Software Engineering Approach to HCI.

The SE 2004 Volume identifies the following as suggested assignments and laboratories [4]: evaluation of user interfaces using heuristic evaluation, evaluation using videotaped observations, prototyping of interfaces, writers workshops for critiquing prototypes, and construction of significant user interfaces using rapid prototyping. The process of developing user interfaces is highly iterative and requires a great deal of interaction with a user. In the traditional model, where homework and laboratories are performed outside of class, getting the amount of feedback necessary to properly model the UI design activity can be a challenge. By using the inverted classroom model, the contact hours can be used to thoroughly involve students in a collaborative and iterative experience that involves users and other participants.

SE 311 Software Design and Architecture.

The SE 2004 Volume identifies the following topics as relevant to this course: design patterns, study of middleware, examination of case studies, application of metrics, and study of reverse and re-engineering [4]. In regards to learning activities, the SE 2004 volume is sparse. However, given the above topics, the course contact hours in an inverted classroom model can focus on short, one hour practice activities or longer case study activities that reinforce knowledge in the above areas. Software design activities, especially the creation of models, benefit greatly from iteration and reinforcement. The assignments and activities provided during class contact hours in this course can also include the analysis of requirements in order to construct software architectures, the use of modeling and different modeling tools, and analysis and review of design models.

SE 321 Software Quality Assurance and Testing.

Software testing is a significant activity in the software development process. The SE 2004 Volume identifies the use of automated tools, practice in testing a variety of systems, application of different testing techniques, and use of inspections as the primary laboratory and assignment activities for this course [4]. By using the inverted classroom model, contact hours can be devoted solely to the activity of testing software or on the application of alternative quality assurance techniques, such as design reviews, inspections, or formal analysis. A large number open-source software projects that are available for use in such an activity, providing the opportunity for a rich experience. The in-class contact hours provide the opportunity to perform all of these course activities in a collaborative manner.

SE 322 Software Requirements Analysis.

The SE 2004 Volume identifies the following activities and labs as being relevant for the requirements activity [4]: construction of requirements, analysis of systems to determine qualities and to reverse engineer requirements, interviewing users, use of tools for managing requirements, modeling of requirements with UML, and resolving feature interactions. An important aspect of the requirements activity is the interaction with customers and users to determine the desired behaviors and applicable constraints for the system to be developed. Using inversion provides an opportunity to regularly schedule contact with customers or users. The dedicated contact hours also provides the instructors with the opportunity to demonstrate various aspects of the requirements phase through role playing or other activities that model the practice. In addition, the contact hours can be used to provide students with experience in using the tools necessary to manage and model requirements.

SE 323 Software Project Management.

The SE 2004 Volume identifies a number of activities that are appropriate for laboratories and assignments in the SE 323 course [4]. These include gaining experience using software project management tools, creating cost estimates for projects, evaluating software licenses, and developing project and configuration management plans. Each of these activities would benefit from the collaborative experience provided by inversion of the classroom.

		Learning Activities														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	The student can design applications that incorporate business processes as a fundamental driver for partitioning															X
2	The student can identify and describe the primary issues related to the use of service-oriented architectures and service-oriented									X						
3	The student can develop web services using C# and the ASP.NET platform															
3a	The student can create C# programs										X	X	X	X	X	
3b	The student can create web services using C#													X	X	
3c	The student can create C# applications that utilize web services										X	X				
4	The student can develop web services using Java and the Apache Axis platform															
4a	The student can create Java services							X	X							
4b	The student can create Java applications that utilize web services			X	X	X	X	X								
5	The student will be able to identify the purpose of various technologies in the web services stack including XML, SOAP, WSDL, and WSDL-X									X						
5a	The student can create an XML schema		X													
5b	The student can create an XML document		X													
5c	The student can identify and recognize different aspects of an XML SOAP message			X												
5d	The student can identify and recognize the relationship between SOAP and WSDL				X											
5e	The student can identify and recognize different aspects of an XML WSDL document				X											
5f	The student can describe different quality attributes related to WSDL/SOAP and REST									X						

Table 3: Outcomes Matrix for Pilot Course

SE 400 Software Engineering Capstone.

Contact hours for the capstone course should be entirely devoted to project activities including the holding of meetings, construction of software development artifacts, software development, and testing. The lecture component of the capstone can then be devoted to the delivery of special topic content or other information that is perhaps relevant to the completion of the assigned projects.

SE 4xx Special Topics in Software Engineering.

Special topics courses in a software engineering program can vary widely depending on the focus of various instructors. Later in this paper describe our experiences with a special topics course on service-oriented computing and web services.

5. PRELIMINARY RESULTS

To date we have applied the use of the inverted classroom in one computing course at Miami University. The first course to undergo the piloting of the inverted classroom for computing was a special topics course on service-oriented architecture (SOA) and web services [10]. As of the writing of this paper, two more courses are in progress using this model, a CS101I equivalent course entitled “Fundamentals of Programming and Problem Solving” and a CS103I equivalent entitled “Data Abstractions and Data Structures”. Two more courses are scheduled for the Spring Semester (a repeat of the SOA course and a repeat of the data structures course).

5.1 Service Oriented Architecture

The offering of this course was the first to use the inverted classroom model for computing at Miami University. The podcasted lecture materials for this course consisted of approximately sixty-five (65) separate podcast episodes ranging in duration of just a few minutes to approximately fifty (50) minutes. The lecture materials consisted of video blogs, Powerpoint presentations with voice overs, and screencasts showing examples of using various software engineering tools including sessions with Eclipse, the Eclipse debugger, Visual Studio, and other tools relevant to the development of web services.

Table 3 shows a course outcomes matrix that maps course outcomes to the learning activities for the course. The gray bars indicate top level outcomes for which there are refined outcomes. For instance, outcome four (4 The student can develop web services using Java and the Apache Axis platform) has been refined to have two child outcomes (4a The student can create Java services and 4b The student can create Java applications that utilize web services). The course consisted of fifteen (15) learning activities that were often 1 to 2 contact hours in duration. The table shows which course outcomes were covered by various learning activities.

The course also included a significant project which students proposed at midterm and completed by the end of the semester. Student response to the course was overwhelmingly positive, especially in regards to the use of the inverted classroom model.

Table 4 shows an indirect assessment of one of the outcomes for the course. While there is no long-term data from which to compare these results with, it does show that at

the very least from the standpoint of the learners, that an outcome of the course was being met. In the table, the left side of each column shows number of responses at the beginning of the semester and the right side of each column shows the responses at the end of the semester. The data shows that from the viewpoint of the students, that some level of learning occurred, thus moving the responses from largely “disagree” to “strongly agree”.

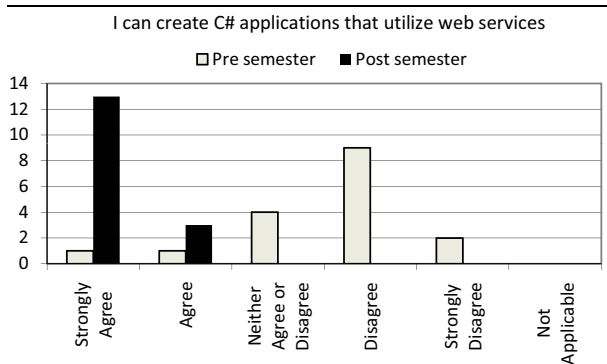


Table 4: Outcome 3c Assessment

A peer review of the course from a colleague that attended the course for the entire semester yielded some of the following comments:

- *The instructor used a variety of class activities, as appropriate. The primary in-class activity was the in-class assignments which is appropriate for the inverted classroom model.*
- *The visual presentation was clearly visible and legible. The software demonstration podcasts were particularly effective learning tools. I often found myself going back over these podcasts to complete an in-class assignment or to pick up a point that I had glossed over. I may try to do something similar to this the next time I need to introduce a piece of software, as students tend to tune out during software demonstrations in class.*

A few comments regarding certain issues with the inverted classroom were as follows:

- *The instructor encouraged student participation and gave students time to respond. Students had the opportunity to ask questions about the podcasts at the beginning of each class. This did not happen very often. I am not sure why it happened this way. Jerry seemed very open to questions and very non-threatening when a student asked a question. There were some questions about assignments, but very few about lecture material. Perhaps the inverted classroom model elicits fewer questions due to the time gap between viewing the podcast material and seeing the instructor or perhaps the students did not have time to view the podcasts.*
- *The students seemed to be actively engaged in the lesson. Certainly the students were actively engaged in the in-class assignment which is one of the beauties of the inverted classroom model. I’m not sure everyone was engaged during the introduction at the beginning.*

Some students were working at their laptops and there was a small group of students in the back that seemed to carry on their own conversation. Perhaps as Jerry perfects this style of teaching, the in-class assignments can be distributed ahead of time and students can get to work more quickly.

5.2 Data Structures

In the Fall Semester of 2007 we are in process of piloting the use of the inverted classroom for the data abstraction and data structures course. The course is using the same model as the one used for the web services course in the sense that podcasting is being used as the primary medium for delivering course content, and that the learning activities are performed during the course contact hours. The course consists of two (2) sections with a total of three (3) contact hours per week for each. Enrollment in the course consists of approximately forty (40) students. The philosophy of the course is based on the notion that repetition and reinforcement when learning programming is paramount. As such, rather than just a handful of programming projects, a greater volume of short programming assignments are given in class. For this course, approximately twenty-four (24) programming assignments are planned in addition to one significant programming project.

5.3 Fundamentals of Programming

For the Fall Semester of 2007, three sections of Fundamentals of Programming and Problem Solving at Miami University are using the inverted classroom to deliver lecture content to over eighty (80) students. Applying the inverted classroom to this class is significant since we are reaching a large number of students in their first programming class. Fundamentals of Programming and Problem Solving is the first course for our computer science majors, minors, and is taken by students from other majors such as physics, mathematics, and management information systems.

In this particular pilot section, we are using a hybrid technique in applying the inverted classroom. Our three (3) contact hours each week are divided up into one (1) hour of lecture time and two (2) hours of lab time. The lecture hour is used as a time to review issues from the previous week, discuss issues with the current weekly programming assignment, and to preview the material in the upcoming podcasts. For this course, our podcast lectures are targeted towards specific topics and are purposely kept short in length (15 to 30 minutes). This allows students to digest one topic at a time, and apply this knowledge in the lab.

In previous offerings of this course, students would complete eight (8) programming assignments (each two weeks in length). This schedule forced multiple topics to be introduced into each programming assignment, often causing confusion about when and how to apply each concept. For the current semester, we have planned for twenty-nine (29) programming assignments. Fourteen of these assignments are shorter lab assignments that are intended to be completed in a single one (1) hour lab session. Each of these assignments introduces exactly one new concept into the body of programming knowledge. The remaining contact hour is used to allow students to begin working on the larger programming assignment for the week. These programming assignments build on the concept introduced in the weekly lab session, incorporate concepts from previous weeks, and sometimes

introduce a second concept for the week. This gradual introduction to programming concepts and constructs allows students to achieve success rapidly. The use of CSCW's [15] integrated automatic grading allows students to receive instant feedback on the functionality of their code and the time spent working the lab with the instructor present helps to set students in the correct direction on the project.

In addition to the introduction of the inverted classroom to this course, we have simultaneously introduced pair programming into the course. After ensuring that all students individually could work in the programming environment and successfully submit programs for grading, all students were paired (with groups of three in sections with odd enrollment numbers). The use of pair programming has been shown to increase student confidence in their programs, to make programming more enjoyable experience, and to aid in student learning simply by encouraging greater participation in the homework process [17]. A recent study has shown that pair programming has benefits for all students, and even more beneficial for women in computing-related majors in terms of confidence and retention [18]. As additional motivation, interviews with students have shown that students view pair programming as beneficial in their learning to program [19].

6. CONCLUSIONS AND FUTURE WORK

At Miami University, the *inverted classroom* model of instruction has been used in a variety of fields including economics, marketing, and now computer science. The approach takes advantage of the benefits of both collaborative learning and distance learning while at the same time targeting the millennial student. In this paper, we have presented a model for using the inverted classroom for software engineering related courses and described our experiences in using the inverted classroom on a few pilot courses. At the conclusion of the Fall 2007 Semester, we plan on interviewing students on both our Fundamentals of Programming and Problem Solving and Data Structures and Data Abstraction courses. These interviews will capture the students' assessment of how inverted classroom techniques impact their learning and success in software engineering courses. Future investigations include piloting the inverted classroom model on a few select courses including the SE 201 Introduction to Software Engineering course. In addition, we will be studying the impact of using the inverted classroom on instructor workload as we embark on repeat use of the approach in the Service Oriented Architecture and Web Services Course as well as the Data Structures and Data Abstraction course at Miami University.

7. ACKNOWLEDGEMENTS

Special thanks to Dr. Donald Byrnett who audited and provided the peer review on the pilot course.

8. REFERENCES

- [1] Maureen J. Lage, Glenn J. Platt, and Michael Treglia. Inverting the Classroom: A Gateway to Creating an Inclusive Learning Environment. *Journal of Economic Education*, 31(1):30–43, Winter 2000.
- [2] Gardner Campbell. There's Something in the Air: Podcasting in Education. *EDUCAUSE Review*, 40(6):32–47, November/December 2005.
- [3] iTunesU. http://www.apple.com/education/itunes_u/ (Visited Oct 10, 2007).
- [4] Joint Task Force on Computing Curricula. Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Proposals in Software Engineering. <http://sites.computer.org/ccse> (Visited October 03, 2007), 2004.
- [5] David W. Johnson and Roger T. Johnson. Instructional Goal Structure: Cooperative, Competitive, or Individualistic. *Review of Educational Research*, 44(2):213–240, Spring 1974.
- [6] Suzanne W. Dietrich and Susan D. Urban. A Cooperative Learning Approach to Database Group Projects: Integrating Theory and Practice. *IEEE Transactions on Education*, 41:14, 1998.
- [7] John D. Tvedt, Roseanne Tesoriero, and Kevin A. Gary. The Software Factory: Combining Undergraduate Computer Science and Software Engineering Education. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 633–642. IEEE, 2001.
- [8] Jason L. Frand. The Information-Age Mindset: Changes in Students and Implications for Higher Education. *EDUCAUSE Review*, 35(5):15–24, September–October 2000.
- [9] Joel Foreman. Next-Generation Educational Technology versus the Lecture. *EDUCAUSE Review*, 35(5):12–22, September/October 2003.
- [10] Gerald C. Gannod. WIP: Using podcasting in an inverted classroom. In *Proceedings of the 37th IEEE Frontiers in Education Conference*. IEEE, 2007.
- [11] Profcast. <http://www.profcast.com> (Visited Oct 6, 2007).
- [12] Snapz pro. <http://www.ambrosiasw.com> (Visited Oct 6, 2007).
- [13] ilife. <http://www.apple.com/ilife> (Visited Oct 6, 2007).
- [14] Blackboard. <http://www.blackboard.com/us/index.Bb> (Visited Oct 6, 2007).
- [15] Michael T. Helmick. Integrated online courseware for computer science courses. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 146–150, New York, NY, USA, 2007. ACM Press.
- [16] Joint Task Force on Computing Curricula. Computing Curricula 2001: Computer Science. <http://www.sigcse.org/cc2001> (Visited Oct 6, 2007), December 2001.
- [17] Brian Hanks, Charlie McDowell, David Draper, and Milovan Krnjajic. Program quality with pair programming in cs1. In *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 176–180, New York, NY, USA, 2004. ACM Press.
- [18] Linda L. Werner, Brian Hanks, and Charlie McDowell. Pair-programming helps female computer science students. *J. Educ. Resour. Comput.*, 4(1):4, 2004.
- [19] Beth Simon and Brian Hanks. First year students' impressions of pair programming in cs1. In *ICER '07: Proceedings of the third international workshop on Computing education research*, pages 73–86, New York, NY, USA, 2007. ACM Press.