AC 2009-2077: CUSTOMIZED INSTRUCTION IN A WEB-BASED, FIRST-YEAR CLASS: MAINTAINING PRESENCE AND THE IMPORTANCE OF TRANSITION USING CONTENT-MANAGEMENT TOOLS

Srikanth Tadepalli, University of Texas, Austin

Srikanth Tadepalli is a PhD candidate in Mechanical Engineering at The University of Texas. After recieving his BS in Mechanical Engineering from India, he moved to UT where obtained his MSE in Manufacturing Systems Engineering specializing in Design for Manufacturing. He has worked as a Teaching Assistant and as an Assistant Instructor for the Computers and Programming course over a period of 3 years at The University of Texas at Austin and was awarded "The H. Grady Rylander Longhorn Mechanical Engineering Club Excellence in Teaching" Fellowship award for the years 2003-2004 and 2007-2008. He has also been cited in multiple publications of the "Who's Who" series. His research interests include Similitude and Scaling Theory, System Dynamics, Non-Linear Dimensional Analysis and Rapid Prototyping with specific emphasis in Selective Laser Sintering and applications in Product Design.

Mitch Pryor, University of Texas, Austin

Mitch Pryor graduated with a B.S. in Mechanical Engineering from Southern Methodist University in 1993. After teaching high school for two years, he completed his PhD in 2002 at the University of Texas (UT) at Austin where he now works as a Research Scientist in the Robotics Research Group and teaches in the Mechanical Engineering Department. As a researcher, his efforts have focused on software development for robotic systems. Recent research efforts include human/robot interactions, mobile manipulation, and robotic workcell integration including projects funded by NSF, DARPA, DOE/NNSA, and ONR among others. In the ME Department, he has taught graduate and undergraduate courses in programming, numerical methods, and robotics, as well as co-developed a nuclear automation interdisciplinary graduate program. Additionally, he has received academic development funding to study presence and stability in online PSI courses.

Cameron Booth, University of Texas, Austin

Cameron is a PhD student in the Mechanical Engineering department specializing in Dynamic Systems and Controls, and a recepient of the Thrust Fellowship. Cameron received his undergraduate education from Georgetown University and has worked for about 2 years as a Teaching Assistant for the Computers and Programming course taught at The University of Texas at Austin.

© American Society for Engineering Education, 2009

Customized instruction in a web-based, first-year class: maintaining presence and the importance of transition using content management tools

Abstract

Personalized System of Instruction (PSI) is a learning strategy employed in a situation where common academic goals are achieved through the individual efforts and advancement of a particular user. This strategy is particularly useful in first-year programming classes where students with varied backgrounds are introduced to advanced coding practices. Since high schools do not provide equivalent computer exposure, expecting students from diverse educational and cultural backgrounds to assimilate computers and related technologies at the same rate is unrealistic. PSI offers students the freedom to identify their own individual strengths and weaknesses and therefore ascertain the amount of work they personally must invest to successfully complete the course. However, all registered students need to share a common platform that allows them access to the same set of material and tests. It must also implement the requirement of conditional access to new subjects as students progress. This framework is realized through internet and web-based resources where study material is stored for browsing by the student depending on his/her progress. This paper addresses two often neglected elements in this course format: presence and transition. It is imperative for the teaching staff - that does not regularly interact with students – to maintain an engaging presence within the course material so students are neither overwhelmed nor overlooked. Presence is important to provide continuous assistance and mentoring by engaging students in a way that happens naturally in the classroom setting but must be actively sought in this course format. Also critical is the ability to transition to new content, instructors, and implementations, so that the presence, style and preferred content can be customized and easily refreshed. New course instructors may not be adept at using the current system. Thus the learning curve and time investment must be reduced without a commensurate reduction in implementation creativity and flexibility. In an attempt to address these issues, this paper provides a case and implementation strategy for how web-based instruction can be administered effectively while maintaining presence and easing transition.

Introduction

PSI, or Personalized System of Instruction[1], is a unique educational technique employed in a situation where a baseline mastery of material is accomplished on a per student basis through individually paced learning and advancement of a particular user. The idea of PSI as a learning tool was developed as an alternative to traditional class discourse[1]. Using the rather innovative concept of unsupervised instruction that allows for individual growth through proactive learning[2], PSI is well suited to first-year programming classes where students with widely varying experience, ability to assimilate new material and backgrounds are introduced to advanced coding practices. Since there is little standardization across high schools that provide computer exposure, expecting students from diverse educational and cultural backdrops to master computers and related technologies at the same rate is unrealistic. By recognizing that both mastery and time cannot be kept constant, PSI offers students the freedom to identify their own individual

strengths and weaknesses and therefore ascertain the amount of work they personally must invest to successfully complete the course. Keller[3], Koen[2] and Morita[3] elaborate on the functional and logistical requirements of a PSI class and ways and means of course development. Two key elements to consider when developing a PSI course are *presence* and *transition*.

Presence is a technical term for measuring the interaction or relation between a facilitator and the audience: for example the difference in a live and recorded lecture. Presence is a challenging attribute to quantify. In the classroom it describes a professor's ability to capture and maintain the interest of his or her audience. For web-based Personalized System of Instruction style classes, one of the primary challenges is porting this illusive presence to a web-based environment, where the teaching staff loses significant direct, regular interaction with students. In this context, establishing a mentoring relationship with students is difficult. Thus, developing an identity for the teaching staff that engages the students in a way that they feel neither anonymous nor overwhelmed with new material is crucial. Familiarizing students with the staff and establishing a working relationship between the two camps is the challenge that creating a palpable presence seeks to accomplish.

Going hand-in-hand with presence is the ability for the teaching staff to *transition* between proctors, professors, content, themes and languages (programming and spoken) to adapt to changing learning environments while maintaining the course implementation. The changes could be as simple as the turnover of proctors from semester to semester, or as complex as porting the course to a new programming language with a new professor in a new spoken language. Regardless, the unifying "look and feel" of the site along with updated and accurate content is crucial to maintaining a visible presence with students. When students see visual signals that the teaching staff is working to provide up to date content and interfaces, they feel more engaged. Along with transition between professors, proctors or languages, the site needs to be able to transition into new technologies and to maintain a modern style. This includes creating a mobile version of the site for use with smart phones and taking advantage of technologies such as RSS (Real Simple Syndication) Feeds.

Throughout this paper we will examine the original PSI implementation with an eye towards a better realization in the future. We will discuss both methods for creating a more robust online presence as well as enacting new statistic collection tools. Methods to better measure the impact of various changes along with strategies for simplifying transition between incarnations of the site will be examined using the original PSI implementation as a jumping off point.

Previous Research Efforts

Most of the literature review gravitates towards ensuring that a class, irrespective of the mode of instruction, should be flexible enough to deliver engaging and attentive discourse of material. While instructors can choose their own approach in a traditional class oriented lecture, it is imperative for teaching staff of PSI courses to innovate and impart information at comparable levels of presence as in a conventional class room. Koen[4,5] provided pioneering results emphasizing the need and different options available to ensure that presence is achieved for an "online" instructor. This view was supported by

Morita[5] who implemented the idea on a global scale by producing similar methodology in other parts of the world. Hence, together, they have been able to deliver the instructional material to the globe using the web as a platform across international boundaries. The process of implementation has been similar though experiences from overseas have been slightly different from those of this country, which is quite understandable because of cultural differences. However, the objectives and goals have remained same between the two implementations.

Previous Implementation

The backbone of the original PSI realization was a well organized collection of HTML pages. Augmenting this was a series of PHP scripts accessing a MySQL database to keep track of student progress and to provide randomly assigned tests. For the first few semesters this set up was more than workable. However, as the requirements of the course and the proctors changed more and more pages were being updated, rewritten or replaced. When the class changed professors, almost every single page had to be edited, as there were 50+ references to the original professor in static HTML pages. This constant upkeep had two downsides. It took the teaching staff's time away from helping students, and it resulted in old, tired and often conflicting content with mixed and/or out-of-date styling. Yet the inclusion of personal, course related information created an additional sense of presence that cannot be felt in, say, a textbook alone.

The original site used a variety of techniques to try and establish an online presence. A chat room (see Figure 1), populated by proctors during office hours, as well as a comment interface in the test submission system provided the most direct and focused contact with proctors. When office hours were not in session there was a course email address where students could send queries and comments and receive feedback within 24 hours. A webcam, usually used in concert with the chat room, was also implemented so students could see other people in the office hours, and better gauge the presence and availability of proctors participating in the chat conversation. Student progress charts were also available so a student could gauge their efforts relative to their peers. The site also included short motivational video interviews with professors from the department talking about how programming skills are relevant to upcoming courses and integral in the engineering sciences (see Figure 2).

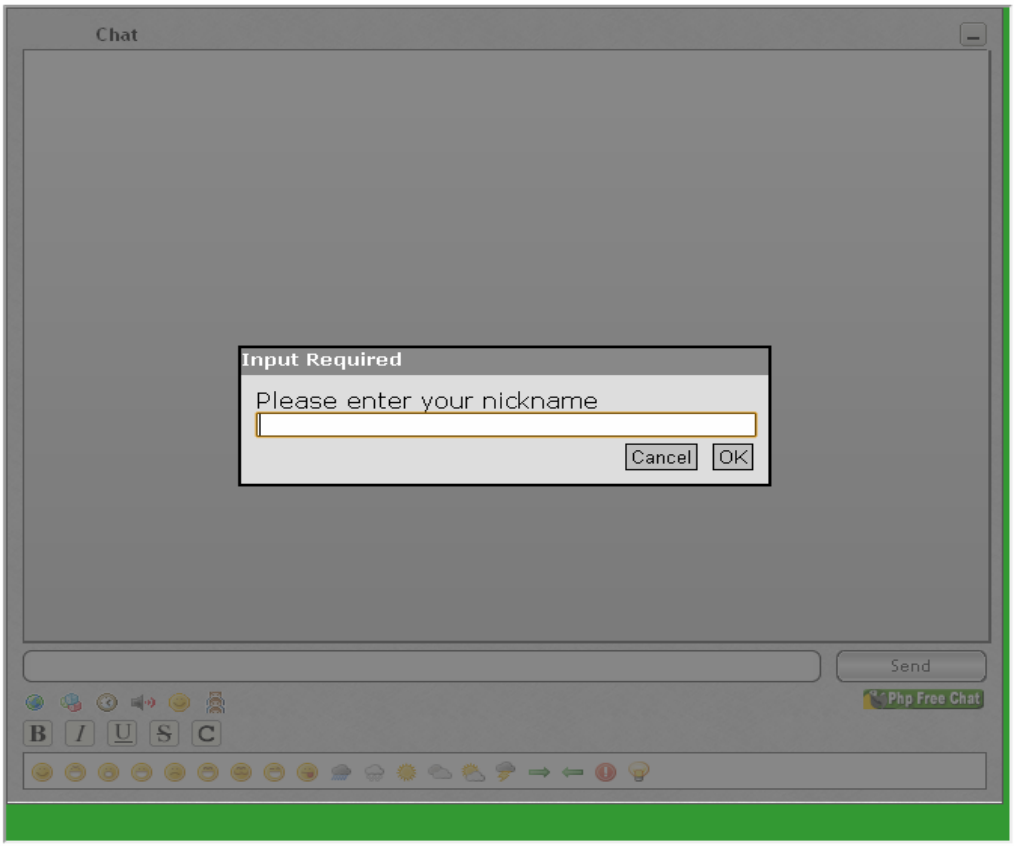


Figure 1: Chat window

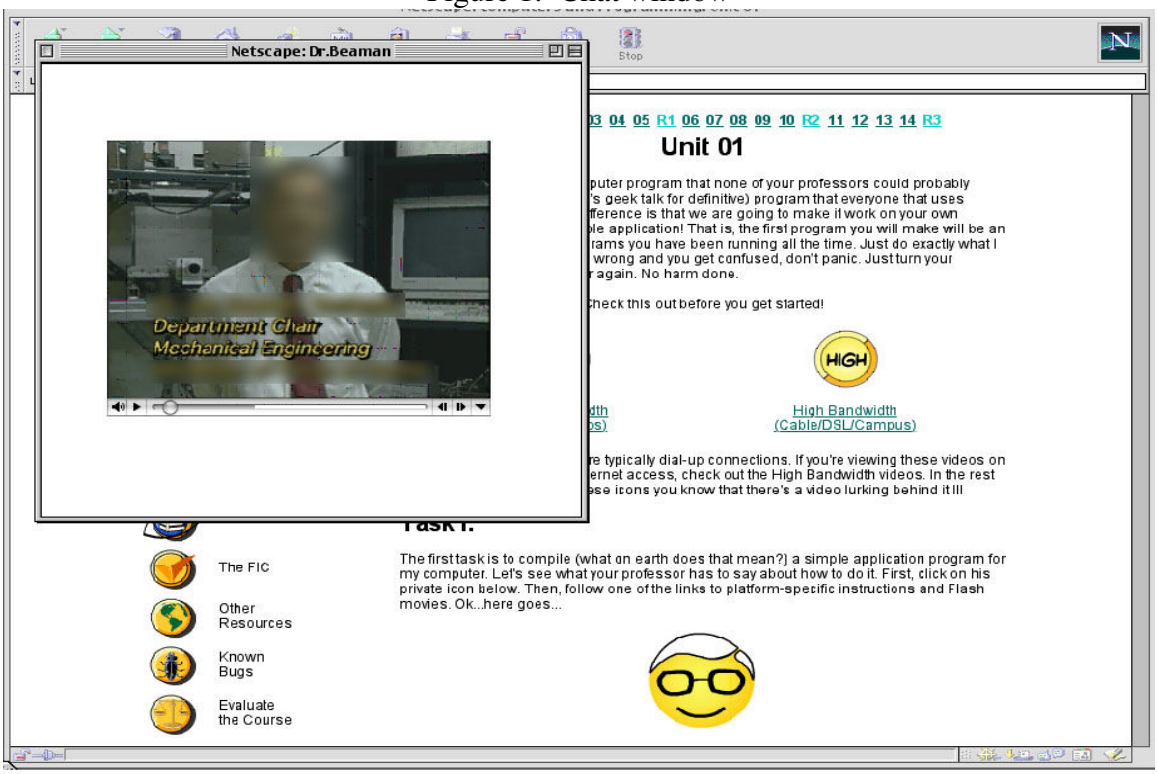


Figure 2: Video describing the class from the ME department's Chair

The downside of some of these presence elements was that they quickly become dated by changes in personnel, antiquation of the technologies discussed, or relevance of the engineering topics discussed. Programming languages change, professors move on, or the movie clips would just look dated. The out-of-date styling, the same content from semester to semester and the difficulties in making rapid, wide-spread updates limited the ability of the site to transition smoothly between different professors, proctors and content. As a result of the lack of frequent and meaningful updates to the website, it gathered a hollow air among both students and staff. We had quickly realized that presence and transition cannot be achieved without the use of modern software tools. Our goal of addressing these two concerns while providing a state-of the-art learning system resulted in an extensive survey of available data base management systems. In an effort to tackle the issue of managing the hundreds of pages while maintaining a coherent, yet easily editable style and content the decision was made to implement a Content Management System (CMS). Further, to reduce costs, we had decided to customize an existing CMS rather than develop our own. This conclusion allowed us to select a CMS of proven capability that is flexible enough to cater to the functional needs of the teaching staff.

Content Management Systems

A Content Management System, or CMS, is a database backed system for maintaining websites. It stores site configuration information in a database and allows an administrative user the ability to create, edit, delete and organize content from a web-based GUI interface. In selecting a CMS we first generated a list of necessary features and then compared our needs with what was available. In the end the key components were:

- Stability
- Compatibility with Language
- Portability
- Easy Maintenance
- Fast Learning Curve
- Free License
- Robust Developers Network
- Extensibility

Fundamentally, implementing a CMS, allows for the developer to easily segregate the independent aspects of the course that may change from semester to semester: instructor, style template, and course content as illustrated in Figure 3.

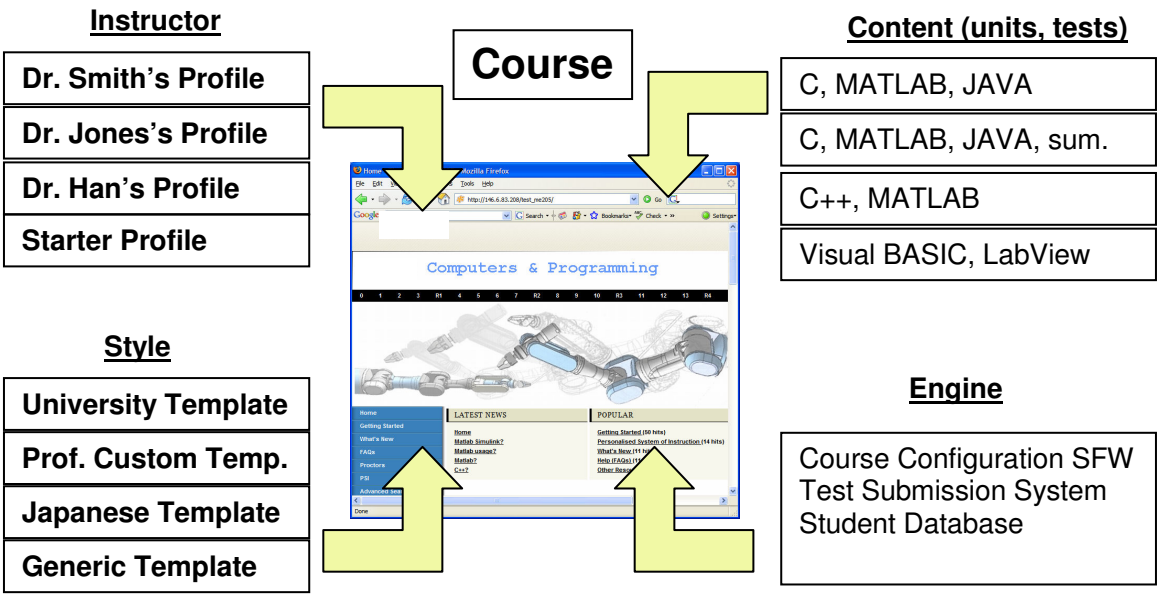


Figure 3: Example segregation of PSI Course components

Starting with the requirement of a free license the front running candidates are Moodle[6], Mambo[7] and Drupal[8]. Creating small test deployments of each system to examine their functionality led us to pick Drupal. Its extensibility, compatibility with the existing PHP structure and its ease of maintenance makes it the best fit for our system. Additionally the developers' network for Drupal has already created a user access extension that allows an administrator to restrict access to content on the site. Along with the access restriction abilities Drupal also has easy to learn user management features.

Requirements	HTML/PHP	Moomba	Mambo	Blackboard	Drupal
Cost	5	5	5	(5)1	5
Stability	4	3	3	(5)1	3
Ease of use (student)	3	5	5	4	5
Ease of use (proctor)	1	3	3	3	5
Ease of use (developer)	2	3	3	1	4
Style Customization	5	3	4	1	3
User access restriction	(5)1	2	2	3	5
Compatibility/Portability	5	4	4	2	4
Total	**25**	**28**	**28**	**16**	**34**

Table 1: Requirements/CMS Candidate Capabilities Matrix

The ability of each product to meet our requirements is ranked from 1 (worst) to 5 (best). Boxes with more than one value indicate a specific ambiguity. For example, currently the university has the Blackboard platform available for all courses, but an individual instructor will obviously not have much input in long term licensing and update decisions. In the case of the HTML/PHP implementations, many desired features can be created from scratch but with great expense and unnecessarily large use of human resources. The worst case scenario is considered in the final rankings.

Asserting Presence

With the new CMS in place we reevaluated the existing presence elements and brainstormed how to improve, replace or augment them. We started by reworking the existing styles and menus to give them a more updated feel. At this point the advantages of the CMS became readily apparent when, after making changes to the layout and style it was reflected through the site (see Figure 4).

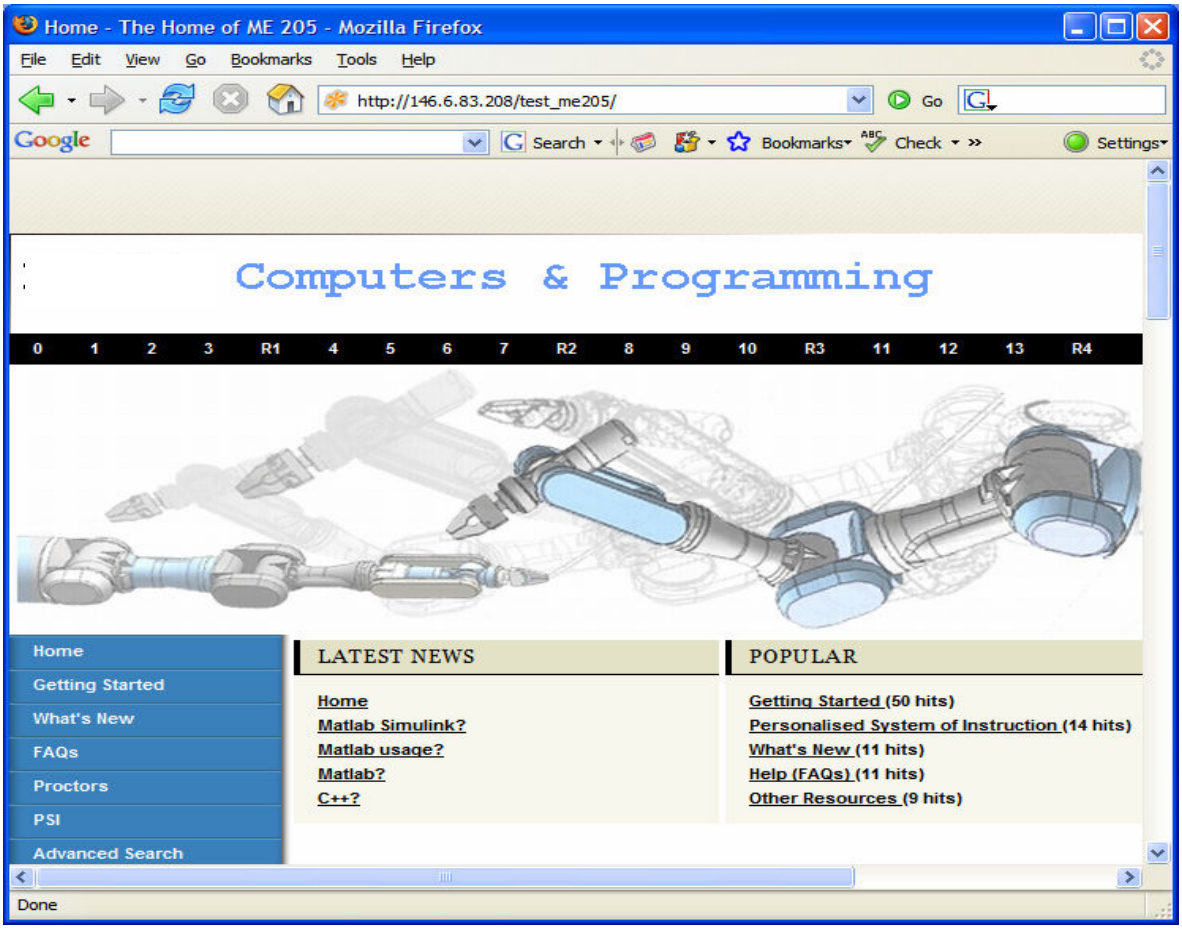


Figure 4: Revised Design Concept for Website

The use of a blog style front page that lists updates to the site, information on updated office hours, deadlines, or small coding tips that are tripping up students was implemented. The chat feature was ported to the new system with a slicker, more polished look. The webcam overlooking office hours was also retained.

Another advantage of the CMS is the ability to use dynamic and pseudo-dynamic content. Initially the student progress reinforcement plots were pseudo-dynamic, and had to be generated by a scheduled maintenance task. With the Drupal implementation however, it was possible to have each individual user's login trigger an update of their respective progress chart, creating a dynamic, up to the minute feedback on course progress (see Figures 5 and 6).

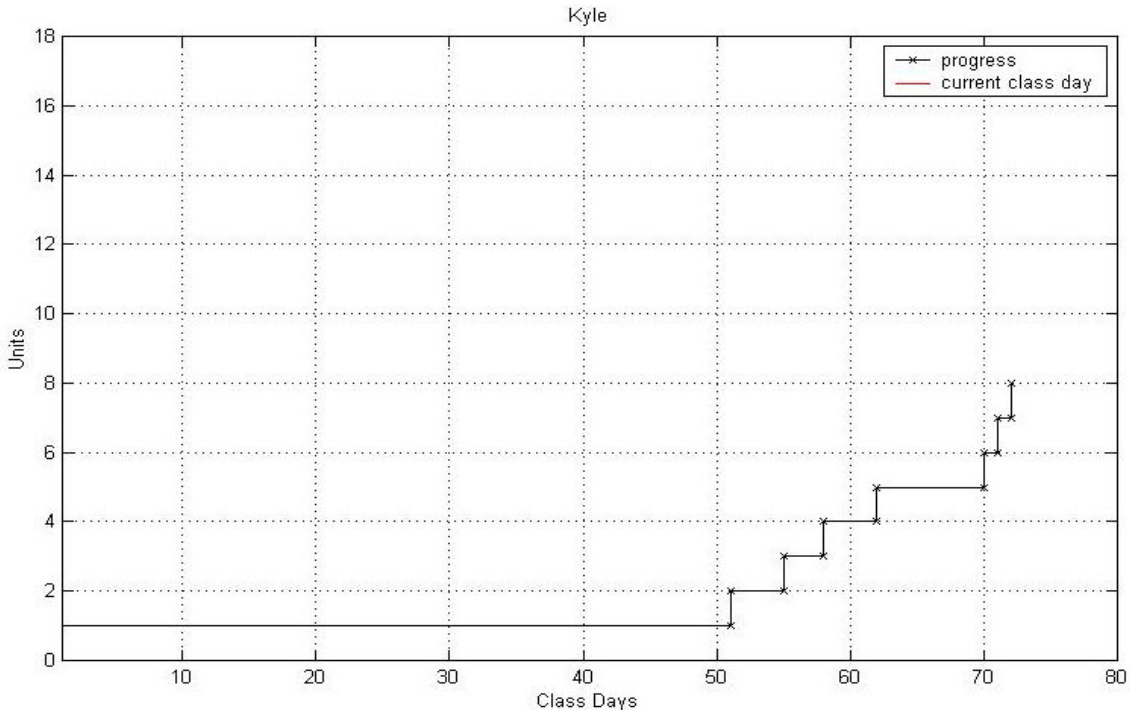


Figure 5: Active individual progress chart

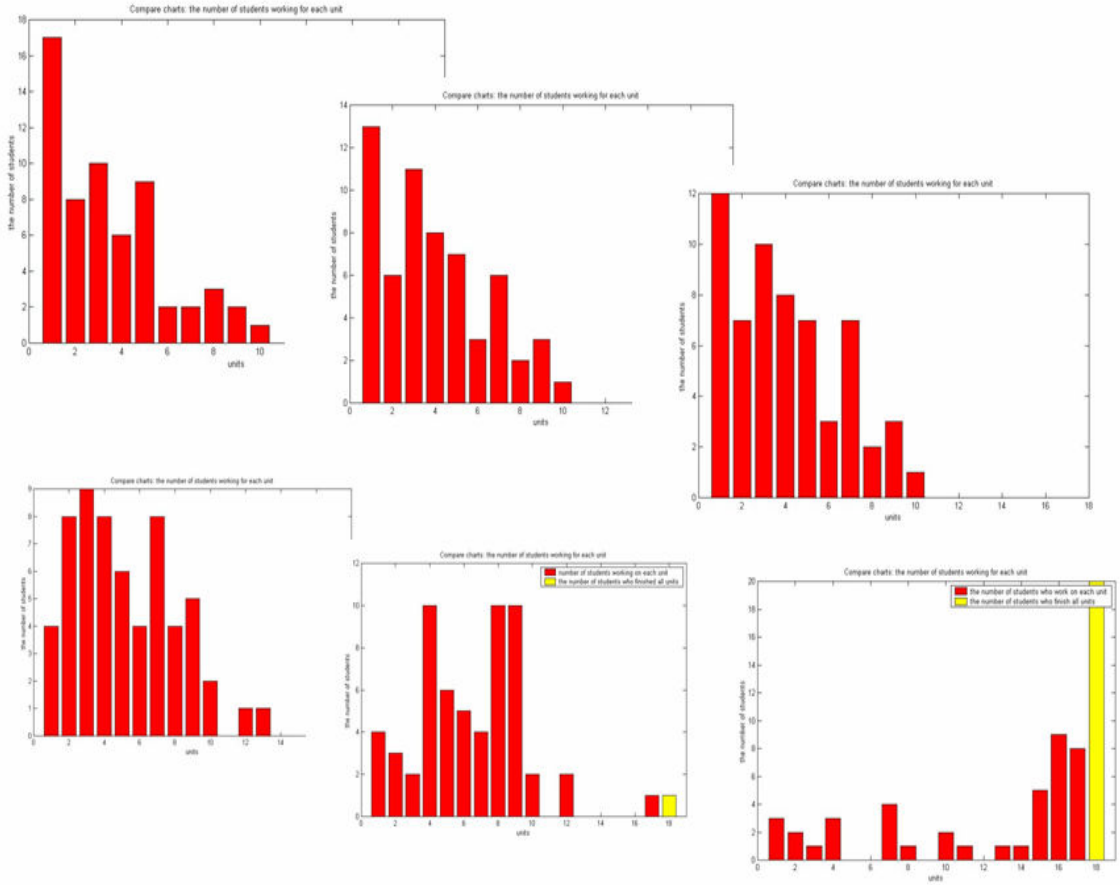


Figure 6: Active class progress charts

Additionally, since content can be published and unpublished, the teaching staff can keep content "in reserve". With reserve content, the staff can generate a presence by indicating specific pending updates, and then easily publishing them incrementally throughout the semester for generating dynamic or updated content. In contrast to the previous system, where static HTML files had to be edited through a secure terminal to the server, the Drupal system also allows browser access to webpages so errors, once spotted, can be immediately corrected.

To further augment our efforts to create a structured and formal instrument for engaging presence, we incorporated a countdown clock on our main page (see Figure 7) both as an informative tool and device for assessing personal time investment. Students are provided continuous updates about impending deadlines thus instigating a sense a personal involvement in the course without excessive external regulation – one of the features of PSI.

Future implementation plans include creating "virtual study groups", where the course can be subdivided into smaller sections so that students can get better acquainted with some of their classmates and can help and encourage each other to complete the course. The concept of using performance based test generation to progress students either more quickly, or more slowly, depending on their demonstration and understanding has also been considered.

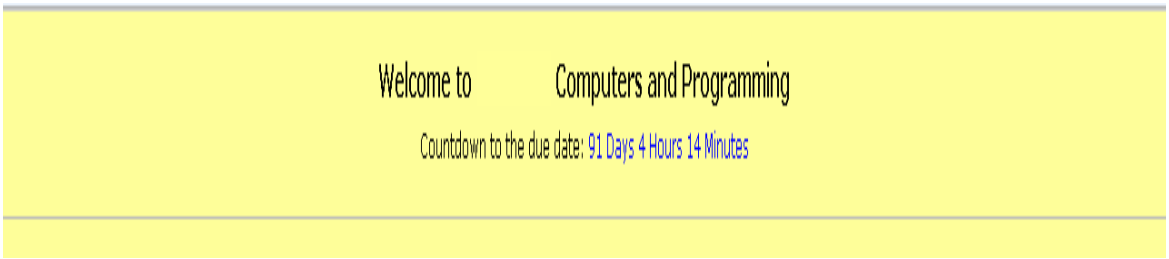
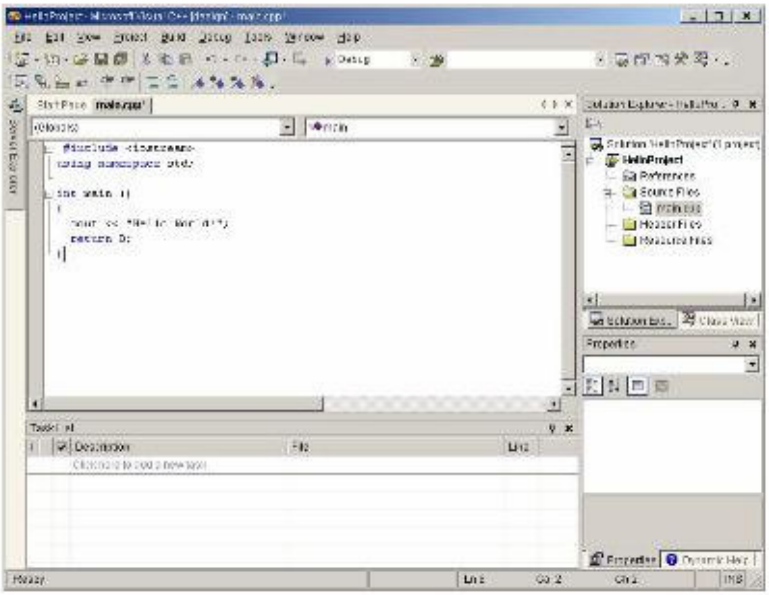


Figure 7: Countdown clock

We have also created impressionable and personable proctor pages, complete with photos and brief bios, which are designed to work in conjunction with a sleek chat window. This allows students to constructively interact with the proctor albeit virtually. This lessens a sense of awe, compensates for lack of physical presence and generates a friendlier ambience while allowing for efficient mentorship. It is a simple yet effective way of telling students that proctors have themselves been novices at programming at some point in their life.

To minimize anxiety and apprehension of first time users, we have also implemented simple demonstrations of coding practices by using screen shots and movies to illustrate the idea creating and compiling a basic program, complete with syntax, structure and output (see Figure 8). This allows students to easily assimilate the abstract idea of generating an error free program from scratch, and ways of compiling and executing the code. This also motivates students to engage in active debugging, a much sought after technique, where we instill the notion that effective learning is possible by understanding

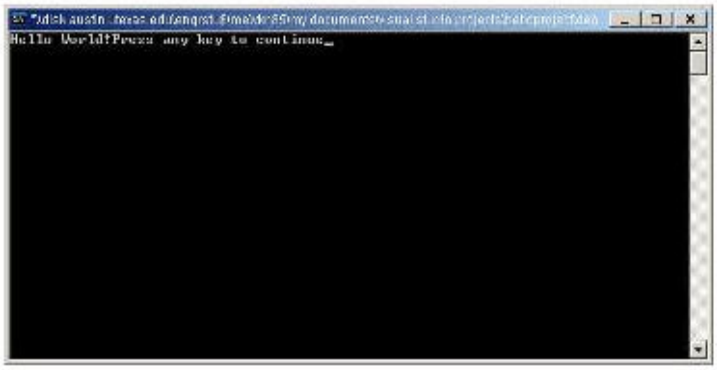
and working on correcting errors rather than asking and observing – a modified Socratic approach.



Click on picture to enlarge.

Are you ready? Here is the big moment! From the menus select **Build** > **Build Solution** If you typed the code in correctly, the 'Output' window should have the word 'Succeeded' print out for you. The Output Window is where the computer let's you know what it thinks of your program.

If you saw that word 'Succeeded', you are ready to run your program. If not, there may be an error or two in your code. Double check your typing and check the output window for clues. As soon as you are ready select **Debug** > **Start witout Debugging** or press Ctrl-F5. And.....



Click on picture to enlarge.

You have just created your first 'Hello World!' Program! Pat yourself on the back. Crack your knuckles and get ready, because hello is only the first word in a conversation!

Figure 8: Active debugging illustration

To further buttress our philosophy of doing by learning, we have created webpages (see Figure 9) informing students about other online resources available for reference including search engines, sample programs, beginner books and templates.

Welcome to Computers and Programming

Countdown to the due date: 91 Days 2 Hours 4 Minutes

Home

Other Resources

Other Resources

As you might suspect, there are other resources available for each of these topics. We hope the materials provided in this course are sufficient to make your learning experience successful, but you may want to examine the following additional resources:

To access the required software for this course (VisualStudio .Net and MatLab), Naviagate to the **Application Server**.

C++

There are tons of good resources for C++. The computer science department of all schools offer short courses in this subject and any college bookstore will provide a wide variety of titles.

Check out the link below for a good tutorial or use Google's directory listing to find other resources.

http://www.cprogramming.com

http://www.java2s.com/Tutorial/Cpp/CatalogCpp.htm

MATLAB

Likewise, there are many tutorials in MATLAB. The following URLs point to learning materials on the web. They were provided by the makers of MATLAB. You may want to check them out.

http://www.rit.edu/~pnveme/Matlab_Course/DEFAULT.HTM

http://www.eece.maine.edu/mm/matweb.html

http://www.indiana.edu/~statmath/math/matlab/

http://spicerack.sr.unh.edu/~mathadm/tutorial/software/matlab/

Figure 9: Availability of other online tools

Despite all the advances and incorporation of technology, we also rely on old fashioned email to inform students of any important updates with regards to software patches, and updates to be installed apart from course related information. The teaching staff also recognizes the fact that some face time is mandated for counseling students who are falling behind or dealing with other concerns, which has been duly accommodated with additional office hours and meeting times. This personable approach has led to the development of a cordial learning environment where we have asserted presence of the teacher without the extravagance of professional counseling. Additional features and policies that have helped boost presence and bolster innovative education include dynamic content updates, timely responses, high resolution status bars, virtual study groups, performance based test generation, supplementary technology introduction.

Transition

Bearing in mind that transition also plays a significant role in engaging student attention, we have migrated from Code Warrior, an unsupported Integrated Development Environment (IDE) to Microsoft's .NET™ (see Figure 10), a more modern and easily accessible software package that is also available to students at subsidized rates. This

allows students to use modern tools on any machine and platform without the worry of patches and fixes.

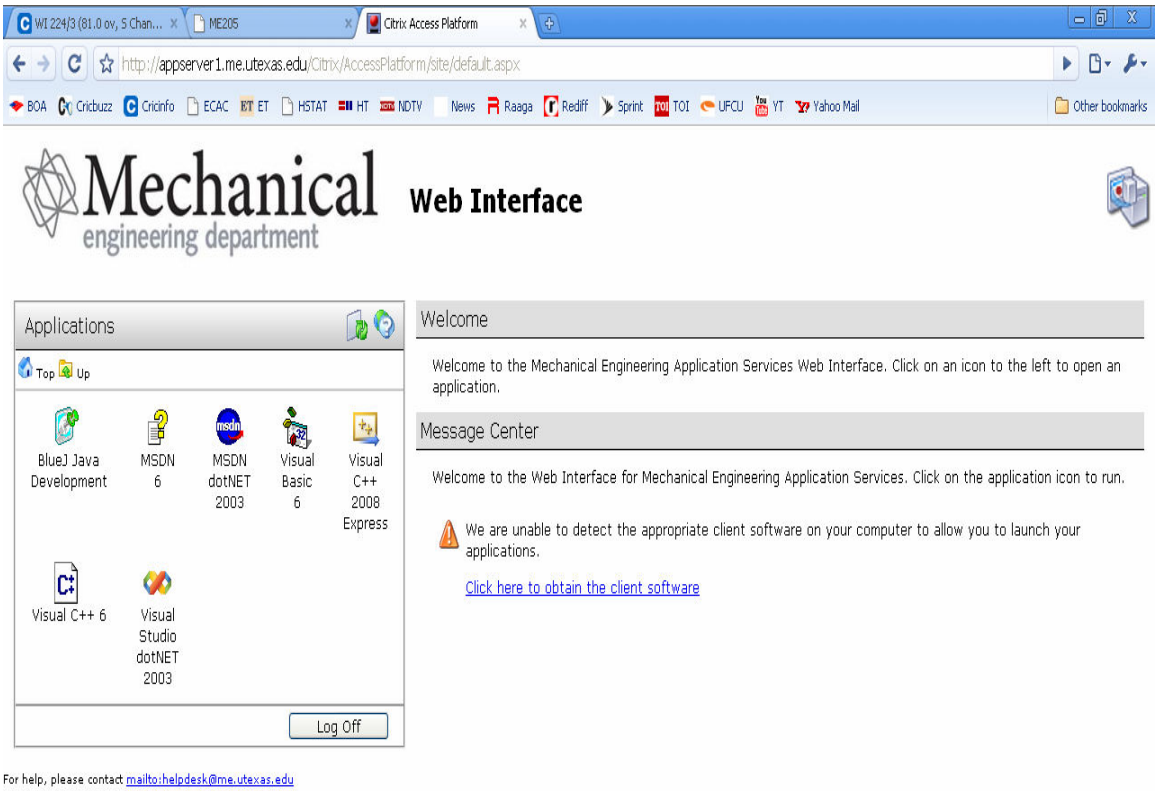


Figure 10: Migration to modern software packages

But perhaps our most important contribution has been the illustration of programming in real life scenarios and applications in engineering systems. Using a variety of software tools, we have shown students how robots can be programmed to deal with issues such as intricate surgeries and hazard management (see Figure 11). This allows students to visually recognize the effectiveness of programming thus realizing the justification of coding in mechanical engineering study. Hence, the perception of programming for students has changed from a fancy and vague concept to a more realistic and doable process. We have thus transitioned from static and abstract programming practice to dynamic and application oriented study using teaching tools to facilitate students to learn by doing, learn by making mistakes, and learn by understanding rather than by methods of observation and rote, which are not pragmatic in learning software.

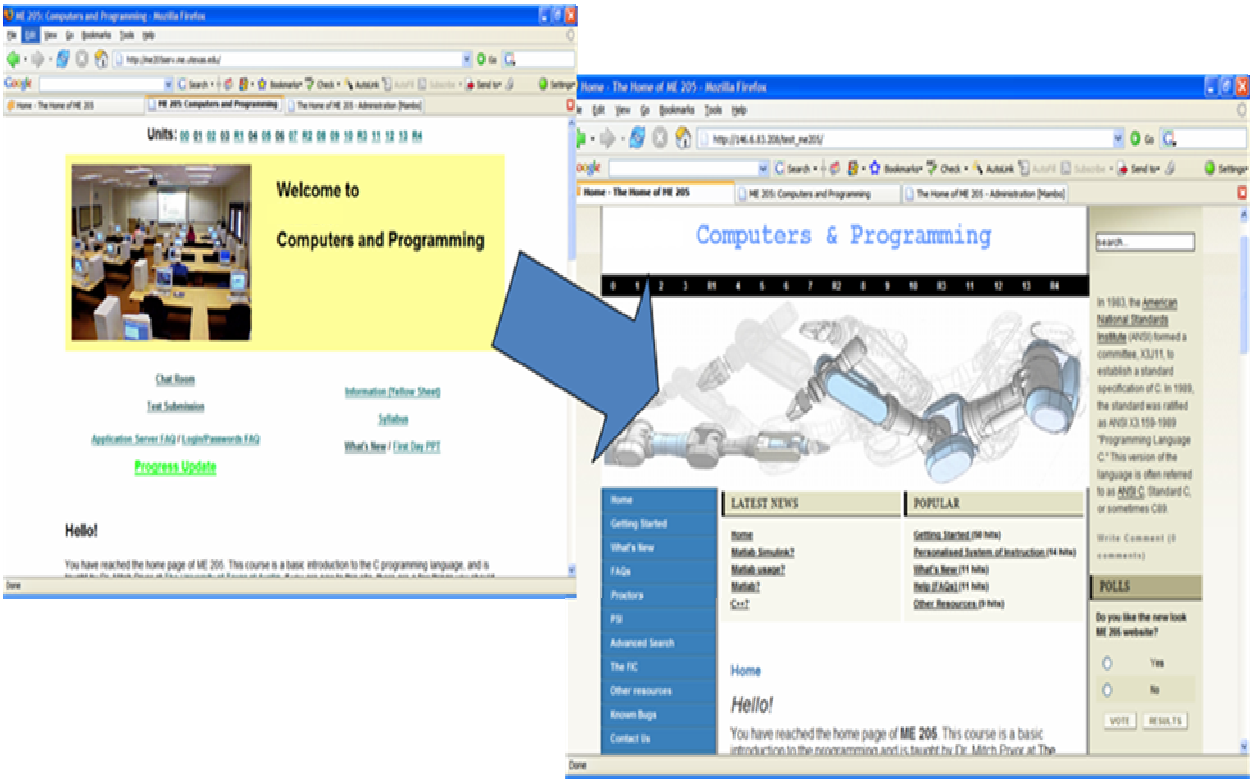


Figure 11: Illustration of programming

Conclusions and Future Work

Student performance and outcome data collection has been implemented for the current semester that can be compared to data for the previous 5 years. Future work will draw on the data from this class in comparison and contrast to past implementations of the class. Like with all novel and innovative concepts, we have initiated a process of measuring the impact of presence using student evaluations, interest metrics (such as procrastination), number of sessions, time spent in sessions, time spent on web site, etc.). We are also in the process of new implementations of statistic collection such as number of repeat students, impacts of presence tools on short term progress, site visits and hits, etc.).

Preliminary data[9] suggests that as content becomes stale students tend to procrastinate and this is a detrimental effect on pass rates and GPA. Thus presence and transition are vital for increased student interaction.

Bibliography

1. Keller, F.S., 1968, "Goodbye,Teacher… ," *J. Appl. Behavior Analysis*, Vol. 1, pp. 79-89, 1968.

2. 4. Koen, B.V., "A Comparison of Conventional and Self-Paced Web-Based Courses: A Theoretical Analysis," *Annual Mtg of American Society for Engineering Education*, Montreal, QC, Canada, June, 2002.

3. Morita, Y., et.al., "Implementation of an International Web-Based PSI Course: A Case Study ," *ASEE/IEEE Frontiers in Education Conference*, San Diego, CA, October, 2006.

4. Koen, B.V., "On the Importance of 'Presence' in a Web-Based Course," *ASEE/IEEE Frontiers in Education Conference*, Boston, MA, November, 2002.

5. Koen, B.V., "Creating a Sense of 'Presence' in a Web-Based PSI Course: The Search for Mark Hopkins," *IEEE Transactions on Education*, Vol. 48, No 4, November, 2005.

6. Moodle Content Management System, www.moodle.com

7. Mambo Content Management System, www.mamboserver.com

8. Drupal Content Management System, www.drupal.org

9. Tadepalli, S., et.al., "Evaluating Academic Procrastination in a Personalized System of Instruction based Curriculum", 116th *ASEE Annual Conference & Exposition*, Austin, TX, June, 2009. (*submitted*)