Tablet PCs and Robots: Technology as Platform and Motivator for Explorations in Collaboration

Michael Wainer, Henry Hexmoor, and Marshall Riley

Computer Science Department, Southern Illinois University, Carbondale, IL, 62901, USA
{wainer, hexmoor, mriley}@cs.siu.edu

ABSTRACT

It is widely recognized that successful software developers and researchers need to be technically competent as well as effective team collaborators. This paper describes how technology (tablet PCs, and mobile robots) was used to focus and enhance opportunities to build collaboration and team skills. Tablet PCs were used to foster social skills within a software development course. The product produced, at the request of a robotics researcher, was an application to simulate robot navigation. The robot navigation application enables explorations in human robotic interaction. The introduction of these technologies has served as a focus to help teach and motivate collaboration skills.

KEYWORDS: Multiagent Systems, Collaboration, CSCW, Pedagogy

1. INTRODUCTION

Computer science students have typically gone through almost all of their computer science curriculum warned to do their own individual work. As they enter upper-level courses they may be assigned team projects but are often left to themselves to divide work within a team. Often the projects they are assigned are fairly small and well defined to test a student's understanding in a specific area. The student may see little connection between the projects they work on and "real" software. Similarly, there may seem to be little connection between the class assignments and what occurs in research or in the industry.

Our goal is to break down these barriers by engaging our students more actively in courses and research and by promoting more hands on collaborative experiences. More than two decades ago Peter Naur advocated that we rethink how we educate software developers to stress an approach, which would have students follow his statement "work on concrete problems under guidance in an active and constructive environment." [7, P. 48] Students should share a common theory of how the system works by effectively communicating and collaborating as they construct their projects.

We agree whole-heartedly with this approach and seek to use technology towards this goal. Technology can provide an enabling platform as well as excitement to promote collaboration. Mobile robots, in addition to having the potential to motivate students [2] have been used in roles which directly facilitate collaboration [8]. We see collaboration as not only an opportunity for students (as developers) to work together but also to illustrate how developers and clients work together as well as how teaching and research work together.

This paper is itself a collaborative effort. Two authors are faculty members. One faculty member was recently awarded an HP technology for teaching grant to enhance an upper-level software development course with tablet PC technology. Another faculty member was interested in obtaining custom software to help with his robotics

research and courses. He acted as a customer for the software development class. Finally, a student from the software development course became an undergraduate robotics research assistant and has continued with the project.

Section 2 describes the software development course and how tablet PC technology was used as a platform to enhance collaboration. The software created in that course (and still evolving) was built over a number of iterations. Working with mobile robots and producing a product of interest for a customer motivated the students. The resulting software as it was used in another class is discussed in section 3. The software was put to use in a robotics course in a way to help build cooperation among student teams as well as an introduction to issues of robot navigation and Human Robot Interaction as explained in section 4. Finally, we summarize our observations and discuss future work in sections 5.

2. TABLET PCS AS PLATFORMS AND MOTIVATOR

For several years we have endeavored to give our software design and development students a "learn by doing" experience based upon agile practices. We have emphasized agile development techniques which have shown much recent success in industry. These methods allow projects to be designed and implemented in an incremental and an iterative fashion [3]. While largely successful for teaching software development, adapting agile methods to an academic environment is not without its problems [9].

Agile software development methodologies compensate for a reduction in rigid bureaucracy by relying heavily on human to human communication skills. Several issues related to the typical class structure and student lives negatively affect student to student communications. Broadly speaking, these issues concern time, space, training, and team skills.

Time: A typical class usually meets 2 or 3 times a week for a total of 3 hours thus the total number of contact hours is minimal. Students are left on their own to schedule meetings with their groups to get most of their work done. It is unfair to expect students to put an inordinate amount of time into a course that is only awarded 3 credit hours.

Space: Students do not have a dedicated work environment. Classrooms are typically designed for passive lectures. Laboratories are designed for students to work rigidly in front of their individual workstation. Rooms which might be suitable for group work do not have computer facilities. Some students (graduates for instance) have special access to labs which others do not.

Training: Students may be lectured on concepts but have difficulty applying those concepts appropriately. Tools may exist to help but without proper guidance the tools may not be used or used incorrectly. Students often resort to familiar techniques or employ a new technique out of individual interest rather than suitability. This problem extends to non-technical aspects as well, such as how to work in a team (listening to other's ideas, etc.)

Team Skills: Students often have little experience in working on a team. Most students make a good attempt to contribute to their team but lack experience and instruction on being a good team contributor. Classes usually contain a diverse group of students (as does the contemporary workplace) and students can benefit from learning and reflecting upon basic social skills and responsibilities to their peers and others. Figure 1 shows a typical student work session.

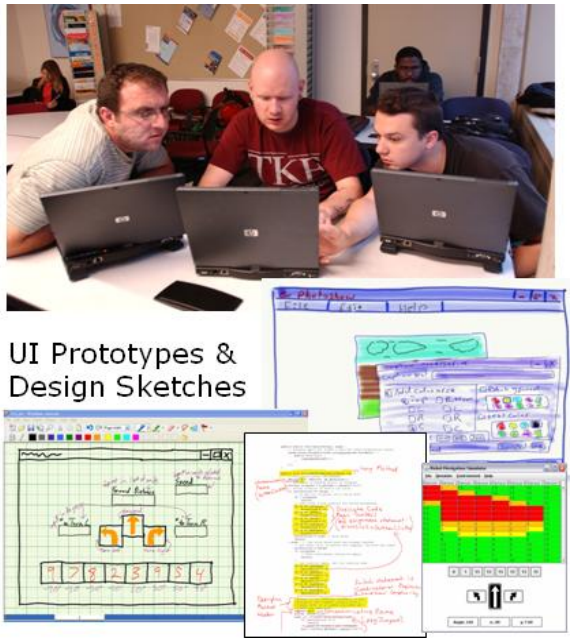

Figure 1. Software Development Students Collaborating and Examples of Group Work Developed Using Tablet PCs

The importance of team skills is repeatedly emphasized by those hiring computer science graduates. In an attempt to improve these skills for students enrolling in software development, a proposal was created to use tablet PC

technology to enhance this course. A recent HP Technology for Teaching grant made it possible for each student in the course to utilize a tablet PC as their development platform. This injection of technology allowed the course to change in several ways to better promote collaboration and team skills.

Space: Each student always had their tablet PC available (they are small and easy to transport). The classroom was changed to a room with furniture and physical layout that supported group work. Wireless capability within the class allowed the computers to be used to their full potential.

Time: The class met 4 times a week, always in the same room. The room was suitable for both lecture and group work activity. More emphasis was provided on allowing students to work together on projects during class time. Because students could easily carry their computers anywhere they could work together more often (utilizing small blocks of time or after labs were closed).

Training: With more time in class for project work, there was more time to answer questions and observe and assist students in using software tools, applying concepts and working together. More experience with group sharing applications such as source code control and wiki software helped to enhance team skills.

Team Skills: Students were instructed from the beginning on the importance of team skills. Almost continuous team projects, practice, and feedback helped to stress the importance of being a responsible team member.

More specifics about how the course was structured is given below. As mentioned earlier, this course produced a simulated robot navigation application. This package, now referred to as Shyster, began to iteratively evolve during the customer project phase of the course. Shyster and user experiences with the software are described in more detail in sections 3 and 4.

Teamwork skills were motivated by group projects, and the use of tablet PCs within a studio-like environment. To get to know each other, and the tablet PCs, an early exercise required students to work together to install software on the tablet PCs. Afterwards, students were assigned a tablet PC for the semester.

Another assignment introduced the XP planning game illustrating the idea of cooperating to iteratively produce and refine a product based upon customer priorities [1] Further group work refined the initial designs from the "planning" game exercise into more detailed low-fidelity prototypes. While students were getting familiar with group work in the low fidelity world, we were also beginning to explore the development tools (installed on their tablet PCs).

Students paired up to understand and later mark-up code samples given to them. The tablet PCs by virtue of their size made it simple for two students to easily share a computer. The digital ink feature of the tablet PCs was intuitive to use for marking-up code samples. Digital ink was also used to refine the designs for the prototype project. (Designs expressed on paper were moved into the computer.)

Student pairs began to modify the sample code (a puzzle game which acted as a practice project) to produce a better product. Communication through code was emphasized. The mark-up exercises helped to show that just getting code to run isn't the ultimate goal. Communication and collaboration of the code itself was emphasized. Students were introduced to and expected to use collaboration software such as the subversion source code control system as well as a wiki to share and post ideas. Refactoring [4] out "smelly" code as well as unit tests (to drive design, check for errors and clarify how the code should be called) and following coding conventions was expected.

The second half of the semester involved working with an external customer to produce the custom software which has now become known as Shyster. Students were expected to explore requirements with the customer and to frequently update the customer on progress by showing working software. The application being created was to be used to model the navigation of a robot through an environment. In collaboration with the customer, the class developed a subset of features to implement. The immediate goal of the application was to provide students (and potential students) with an introduction to the problem of robot navigation.

As an example of customer contact with the class, a series of possible screen sketches were made (using tabletPC digital ink) and posted to the project wiki for customer and class review. The customer gave feedback on which features over the short term had the most value so the developers would know where to focus their efforts. Early essential features were to show sensor readings and to allow the user to direct the robot to move forward and turn incrementally.

Originally, there were some thoughts about the class directly utilizing the physical robots. (P3-DX mobile robot which receives continual sonar readings) The agile software development technique allowed our plans to

adapt. As it turned out, the physical robots became available too late to be incorporated into the class's software release. A more rigid upfront planning process would have wasted a great deal of time designing software which could not have been properly tested or run.

As the project progressed, the tablet PCs facilitated spontaneous small group meetings to collaborate on recent advancement in the design and structure of the program. For instance, when the sonar reading history log was conceived, the tablet PCs aided in communicating possibilities for content and structure prior to adding it to the code base.

The class ended with a product that had contributions from the entire class. Everyone in the class had gained experience working on a team using face to face, written and shared code (through a software repository). While some students made stronger contributions than others, everyone gained a new understanding for the importance of teamwork and collaboration in the production of software. The software, not yet a polished product, demonstrated strong enough potential that, it was decided to continue to improve upon it. The details of the application produced are described next.

3. THE PRODUCT: SHYSTER A ROBOT SIMULATOR

The application produced is a Java program that mimics a P3-DX mobile robot (refer to www.mobilerobots.com). The robot receives continual sonar readings as a user issues incremental, step-wise forward and turn commands. Figure 2 is a screen-shot that shows the sonar view on the left and an overhead view on the right. The overhead view shown in Figure 2 (the right panel) shows the robot in a box canyon, i.e., surrounded by three walls.

A user may introduce environments constructed from line segments acting as obstacles in the simulated environment using a pull down menu. The bottom left circle in the sonar view is a compass depicting the current robot heading direction (the needle faces upwards when the robot is facing up, etc.). Three bottom arrows in the sonar view are available for user mouse clicks. Relative sizes of rectangular boxes in the image depict relative sonar reading amounts. I.e., larger boxes mimic looming obstacles that indicate stronger sonar readings corresponding to closer objects. Each forward click propels the robot forward a unit while turn clicks rotate the robot heading at 30 degrees per command. Locomotion (displacement) is produced only through forward commands. Turn and forward commands are also available through accelerator keys. Each time the robot turns or moves the older sonar readings are redisplayed on the next row providing history of previous readings. The solid yellow-orange lines, below the sonar display area, indicate relative positions of the eight sonar sensors affixed to the body of the robots in a front facing semi-circular pattern. In the overhead view Shyster leaves a trail of robot paths, which aids in navigation and debugging. Backward locomotion is not supported so the robot must be turned about to escape corners.

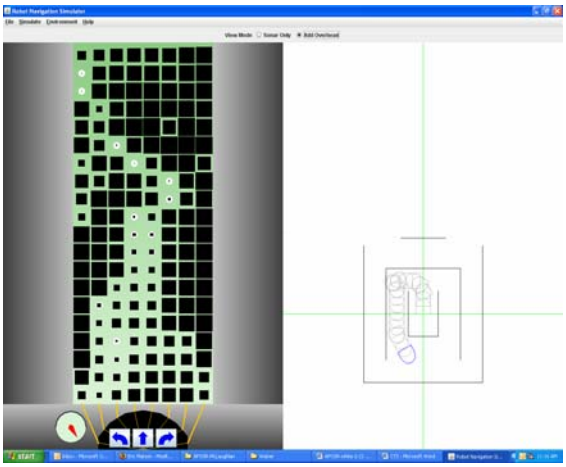

Figure 2. A Snapshot of Our Implemented HRI Interface

4. ROBOTIC CLASSROOM EXPERIENCES

Beyond the class that developed it, Shyster was used in a senior mobile robotics class to present a simulation of a P3DX robot. Everyone in the class was given an individual copy of the software and was observed learning and using the simulation. Shyster displayed a P3DX robot in an open environment on start. Preset environments could be loaded or a unique environment of the user's choice could be created. Students attempted to establish their robot's position within its environment relying solely upon sonar. This proved difficult to do. An additional visualization showing the robot from overhead along with its "foot prints" from previous locations helped students to correlate position within the environment with the sonar readings observed. Custom environments were created with the software allowing the users to see how sonar was influenced in various situations.

After having a chance to experience robot navigation with Shyster, students discussed their thoughts and observations. Many comments focused upon suggesting improvements for the Shyster interface. This itself raises additional opportunities for collaboration with other courses such as Interaction Design and Software

Development. A more structured comparison of Shyster interface alternatives will likely yield a better interface design. The experience and discussions helped to give students insights into the challenges of Human Robot Interaction. Among the comments raised were modifications to the sonar readings such as: shading the feedback squares, showing an outline making clear the maximum reading, or even read-outs as numbers. Other users may have preferred a different mapping of control keys or perhaps a command line interface to specify semi automation control over the robot. Overlaying a grid in the visual map was also proposed to ease plotting walls within the customized environment.

A visual feedback option is our latest feature added to Shyster. It allows the human user to observe the environment in a popup window that receives a feed from a live robot-mounted camera input stream, See Figures 3 and 4. This option adds a degree of realism lacking from sonar-only feedback.

The robotics class continued experimentation with group collaborations beyond Shyster with a role playing scenario. Teams consisting of two people took the roles of either a robot or a human. Human roles were told to give orders to the robot players in both high autonomy and low autonomy modes. In low autonomy, many orders were given more feedback was required before future requests were issued. This produced a blow by blow style of robot control. An example series of commands is "move forward 5 feet, then turn 90 degrees". The main task of the pair was to measure the length and width of the classroom. Students adapted this method and used sonar and bumper sensors to obtain the size of the room. One individual playing the role of the robot was told to "move forward until you hit an object, then stop" and then give feedback of the distance traveled. Low autonomy required waiting for additional feedback and required excessive amounts of management. High autonomy was then attempted. Orders were given such as "Move forward until you hit something, then turn 90 degrees and keep going". This was an attempt to see if the robot actor found length and width of the room without human intervention. Using this method, the individual playing the role of the robot began to get stuck in circles around objects. High autonomy mode was largely efficient, albeit at the expense of more problems of movement and interaction in the world. Low autonomy mode required an excessive amount of management to reach the desired goal. Relatively more factors must be considered in high autonomy. The definitions of high autonomy and low autonomy were well established before the skit, but this skit did provide insight on the advantages and disadvantages of each one. Alternative directions were given to the individuals playing the role of the robot in high autonomy to work around conflicts, such as starting from a better position, watching for areas that may result in the robot getting stuck, and more informed feedback to adjust the situation when necessary.

Figure 3. A Bird's Eye View of the Robot-Mounted Camera Corresponding to the Robot View Shown in Figure 4.

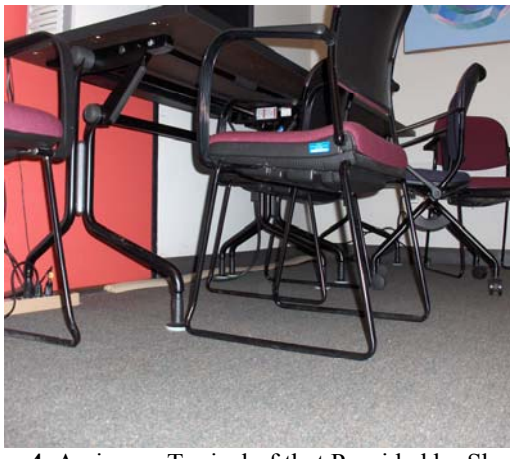

Figure 4. An image Typical of that Provided by Shyster's Robot Camera Feed

5. OBSERVATIONS AND CONCLUSIONS

There were doubts among faculty about whether or not students would responsibly protect and maintain their tablet PCs throughout the semester. This turned out to be a non-issue. It would have helped the students if our department had a convenient place to store the tablet PCs and many students would have found a backpack carrying case more suited to their lifestyles rather than the standard business case carrier. Tablet PCs enable more frequent hands on and collaborative experiences. The value of teamwork and collaboration was appreciated by all. The technology enabled strong support for Naur's notion of having students "work on concrete problems under guidance in an active and constructive environment."

Digital ink was a natural fit for code mark-ups and design sketches. Many students were very comfortable using these machines for note taking as well. When it came to writing code, the keyboard was favored and tablet PCs were often thought of as very portable (with long battery life) notebook computers. To that end, we hope future machines will offer more speed and more screen resolution.

Interestingly the greatest impact of technology may have been the indirect consequences that it enabled. For instance, the class layout could be dynamically reconfigured for lecture, group or individual work. The highly portable nature of the tablet PCs was very valuable in allowing students to physically position themselves to work together (especially for pair programming a technique used in the well known agile methodology, extreme programming [1]. Students could easily pick up or slide their machines around to better work with team-mates (and customers). The flexibility in where we could meet (no computer lab required) meant an easier proximity to the class customer. With more contact time and a more convenient location, it was easier to arrange customer meetings.

Robots, and in recent years, human-robot interaction [5, 6] are active areas of research which most computer science students find intensely interesting. In fact, we have found mobile robot technology so interesting, that students are motivated even in projects which do not involve contact with actual robot hardware. Software simulations and role-playing sessions, are valuable exercises for acquainting students with robotics principles while spurring conversations and teamwork. Thus a small amount of robotics hardware, can serve a larger body of students as motivation for a variety of projects. We hope to use this observation to foster more collaborative experiences between robotics research and exercises undertaken in other courses.

This work has illustrated our experiences in using technology as an effective platform and motivator for collaboration within computer science courses. Tablet PCs, being extremely portable and supporting an intuitive pen based interface, have removed barriers in space and time to enhance team and project skills. Robots, served as a strong motivator for students even in exercises in which hardware was not physically present.

Increasing hands-on and collaboration experiences in the classroom emphasizes active learning but also offers an exciting challenge to instructors: How to pace class material to balance providing students details and specifics against letting them explore and resolve problems with gentle guidance? We plan on exploring this issue further as we expand our use of technology to promote collaboration and teamwork in and among other courses.

ACKNOWLEDGEMENTS

We wish to thank Hewlett-Packard Company for supporting this work through an HP Technology for Teaching award.

REFERENCES

[1] Beck, K. 1999. Embracing change with extreme programming, Computer, Volume 32, Issue 10 (1999), Pages 70-77.

[2] Blank, D. 2006. Viewpoint: Robots make computer science personal by Douglas Blank Communications of the ACM Volume 49, Number 12 (2006), Pages 25-27.

[3] Cockburn, A. 2002. Agile Software Development, Addison-Wesley.

[4] Fowler, M. 1999. Refactoring: Improving the Design of Existing Code, Addison-Wesley.

[5] Hinds, P., Kiesler, S. (Eds), 2002. Distributed Work, The MIT Press.

[6] Kiesler, S., Hinds, P. 2004. Human-robot interaction, Special issue of HUMAN-COMPUTER INTERACTION, 2004, Volume 19, pp. 1–8, Lawrence Erlbaum Associates, Inc.

[7] Naur, P. 1992. Programming as Theory Building, Computing: A Human Activity., pages 37-48 (reprinted from Microprocessing and Microprogramming 15: 253-261, 1985), ACM Press.

[8] Severinson-Eklundh, K. Green, A., and Hüttenrauch, H. 2003. Social and collaborative aspects of interaction with a service robot, In Robotics and Autonomous Systems, Volume 42, Issues 3-4, Pages 223-234, Elsevier.

[9] Wainer, M., Hays, D. 2003. Evolving Software Development Instruction to Support Agile Practices, In Proceedings of Software Engineering Research and Practice, pp.773-9.