TA Groupware[1]

Timothy J. Hickey
Richard Alterman
John Langton
Department of Computer Science
Brandeis University
Waltham, MA 02254
USA

Abstract

In this paper we describe the TA-tool, a same time/different place groupware application developed to support the use of online teaching assistants in large introductory programming courses. It combines features of an editor, ftp client, and instant messenger client to allow students and teaching assistants to inspect, and comment on, the student's code in real time.

1 Introduction

The computing environment on many college campuses has been changing rapidly in the past few years. In particular,

1. There has been a general increase in the size of programming classes over past five years.
2. There has also been a proliferation of high speed internet connections in dorms and homes.
3. Most students today are frequent users of various groupware tools such as instant messaging services, and hence have high expectations for the technology used in classes.

As our programming classes have grown in size, the problem of providing effective Teaching Assistant (TA) support has grown more difficult. One reason is that students no longer work in a central location where a TA can hold office hours. It is inconvenient for a student to physically visit a TA as it requires a break in activity. Finally, because students have familiarity with interacting online (via popular programs like Instant Messenger), the overhead of introducing such technology into the classroom is lessened.

[1]This work was supported by the National Science Foundation under Grant No. EIA-0082393.

In this paper we describe the "TA-tool", a software application which aims to improve TA support given these trends. The tool is a same time/different place groupware application [3] which allows TAs and students to meet online and inspect/debug/discuss the student's code.

2 Previous Work

We consider three possible approaches other than the one we have developed for supporting online TA/student collaboration.

Partially Automated Code Inspection The Icicle project [1] featured a GUI tool for partially automating some of the code inspection tasks in a group programming project. The tool was envisioned to be used on a single computer in the same rooms as the assembled members of the programming team. The scribe would then put annotation on selected lines of the program corresponding to various code review comments (bugs, syntax errors, etc.) As a framework for developing a TA/student collaborative tool, the code inspection model presents some problems. It requires students and TAs to be physically present with one another. It also provides no means for recording the salient features of their interaction (except for videotaping); this means it is difficult to use the interaction between the TA and student as a resource for identifying student difficulties with the material.

Instant Messaging Another approach is to have students and TAs agree to use Instant Messenger. The student could use the file sharing capabilities provided by Instant Messaging (IM) services to send her program to the TA. The TA would add comments to the program and send it back. The student and TA could then chat about the program. An attractive feature of this approach is that most students are very comfortable using

IM. The main drawback is that it requires the student and TA to discuss the program by reference only (e.g. "the 23rd line seems to contain a syntax error"). Moreover, the lack of interactivity would likely lead to problems in communication [4].

Virtual Network Computing As a third approach, one could use a remote terminal sharing technique such as Virtual Network Computing [2] in which the student and TA would both have full control over a single screen (presumably the student's). In this model the TA could highlight segments of the student's code and edit the student's program. Chatting could be done by opening a text editor window and having the student and TA each type into it to simulate a chat. The main disadvantage of this approach is that it makes it too easy for the TA to directly edit the student's program. This is a strong temptation especially when students are having difficulties understanding the assignment. Another disadvantage is that this approach requires using and installing a sophisticated application (VNC).

The TA-tool combines some of the positive features of the approaches described above while avoiding their drawbacks. It uses the idea in Icicle of allowing a group to inspect a program line-by-line, but restricts the group to two people (student and TA), and relaxes the requirement that the group members be in the same room. The TA-tool incorporates many of the familiar features of IM systems (such as buddy lists and online chatting) as well as the fine granularity of the VNC approach, while introducing mechanisms for increasing interactivity though placing some limits on the degree to which the TA can write the student's program.

3 Key features of the TA-Tool

Figure 1 shows two of the TA-Tool windows. The top window belongs to a student, and the bottom window belongs to a TA and is running on some other computer (usually across campus). This tools implements a partial WYSIWIS (What You See Is What I See: [5]) in which the student and TA can change some aspects of the appearance of their window without the same change occurring on their partner's window (e.g. resizing windows, moving the sliding window bars, etc.)

The most important component is the pair of text areas which appear in the bottom-right of the two windows. The left text area contains the student's program and the right contains the TA's comments. These two windows scroll vertically together, so that the TA's comments remain together with the corresponding lines of the student's program. The tool uses a light color to highlight the line containing the most recent change, and uses a darker color for the previous 100 characters. This highlighting allows the student and TA to each see where the other is working. The remote text areas are also read-only, thus the TA cannot directly change the student's program but can only comment on it, line-by-line. Besides being easier to implement this also serves an important pedagogic goal of reinforcing the TA's role as advisor not as partner in the coding.

Directly above these text areas is the chat area. This allows the student and TA to have a conversation on issues that are independent of the program. There is also a group chat window in which everyone in the class can converse at the same time.

On the left side of the window is a "buddy list" which gives links to other users on the system.[1] The TA has a list of students that are waiting to be helped. For the student, the list contains links to TAs. By clicking on a TA the student either joins with the TA in a group code inspection meeting, or is put on the TAs list of waiting students, depending on whether the TA is busy or not. Students or TAs can terminate a session either be clicking on a different buddy or by switching from "online" to "offline" in the choice component below the chat area. Audible cues are given to indicate changes in the buddy lists. This can be helpful to alert a TA to the presence of a new student and vice versa.

The TA-tool has a number of IDE (Integrated Development Environment) tools for scheme programs (which was the main application area), and some other features such as a Font menu, a Help menu, and a File menu (which allows students to store code either locally or on the class server). Another interesting feature of the TA-tool is that it logs every event and the time (in milliseconds) at which it occurred. The log data collected from TA/student interactions can be reviewed using a VCR-like tool [6] with the usual VCR controls (play, forward, reverse, fast forward, pause). This has proved to be quite useful in assessing students' problems and allows TAs to review what students have been working on.

4 Initial Experience

The initial version of this tool has been used in two small summer school courses. Each had an enrollment of about 10 students who were seated in front of their own PC. The room also had a projected computer image. We found that the students' familiarity with Instant Messaging (IM) generated both excitement and familiarity with using the tool. It was distributed as a Java Network Launching Protocol (JNLP) download using Java Web Start which went smoothly. The students found many of the IDE features attractive as well as the integrated access to the remote file server.

[1] This is similar to the IM buddy lists which show the user which of their friends are online at any given time.

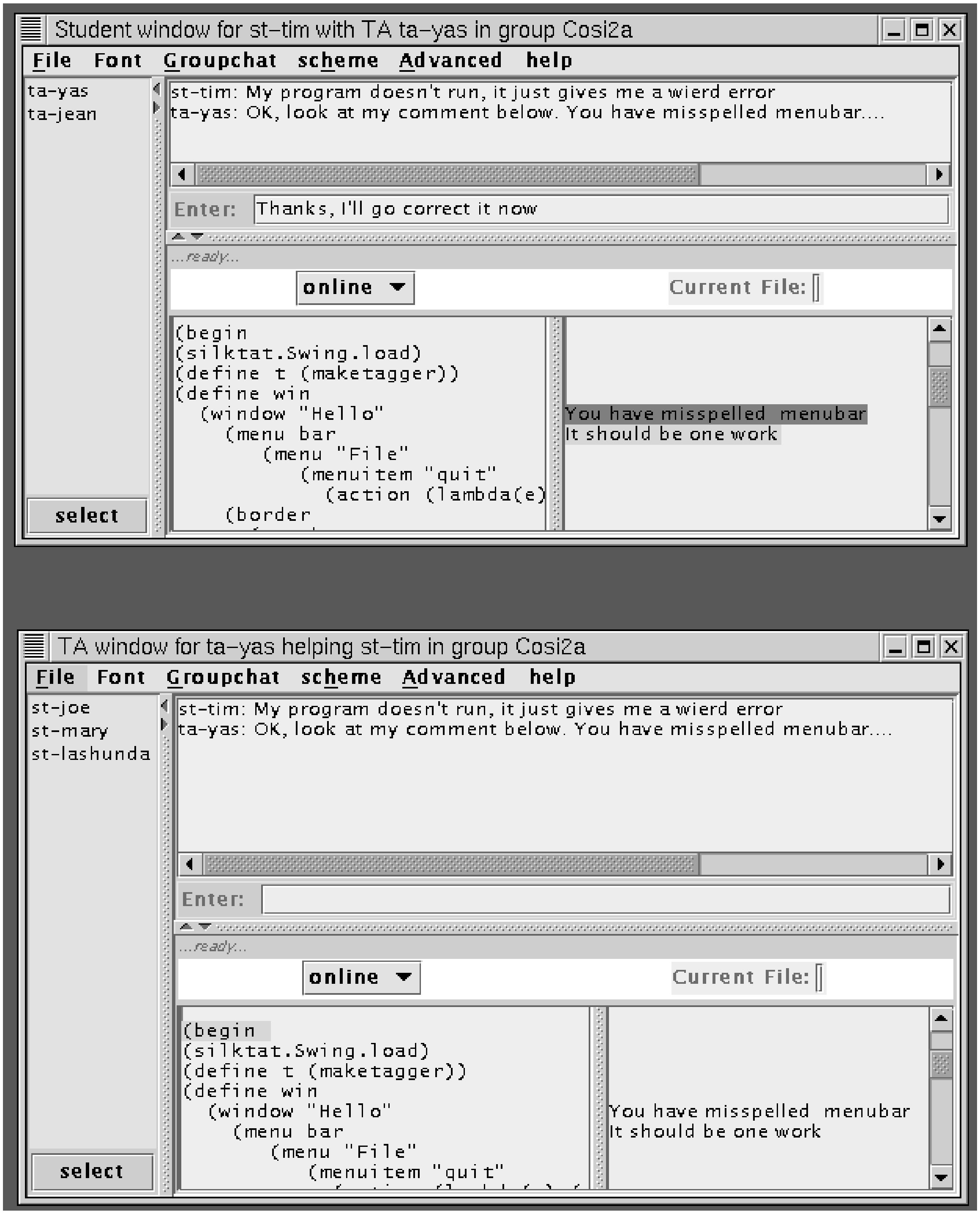

Figure 1: The initial TA-tool

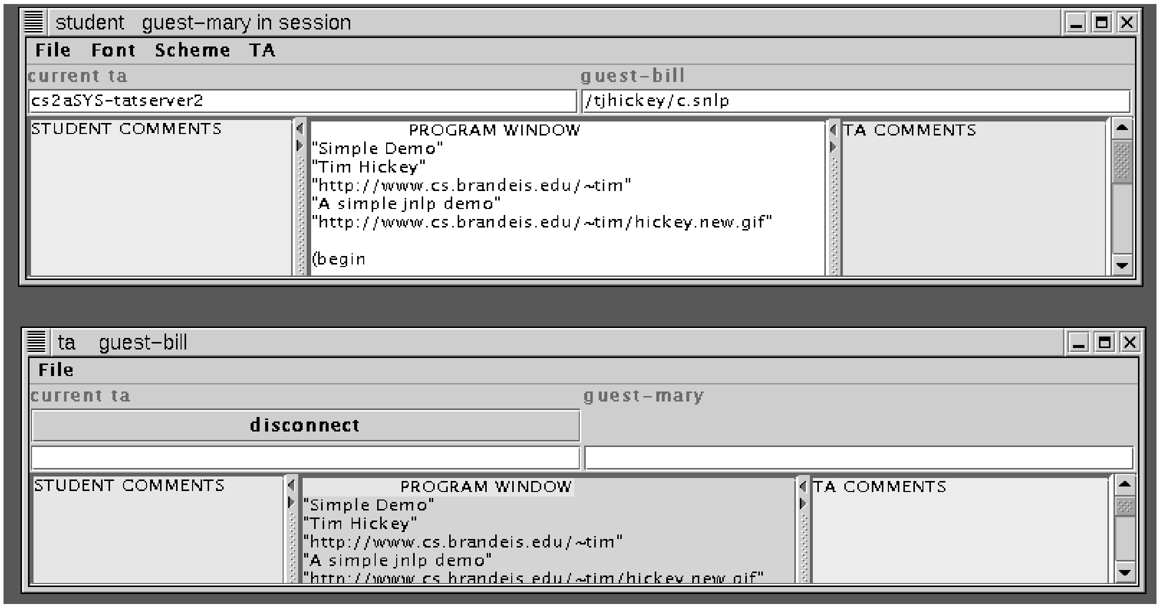

Figure 2: The redesigned TA-Tool

There were however some major drawbacks in this initial design. The primary problem was one of focus. There are three places in which the student and TA can communicate: they can converse through the program/comment text areas (which is awkward for the student as she must write in her code window), in the chat window above, or in the (separate) group chat window. We tried using the tool with one TA opening several windows, one for each student, but found that this sort of many-one interaction did not work well. It further subdivided the TA's attention and made it difficult to maintain an effective conversation.

5 Conclusions and Future Work

We have a second design of the TATool and are deploying it in two large classes (of about 100-200 students each). We will be evaluating this version of the tool over the coming semester.

The new tool (see Fig. 2) adds a comment text area for the student so that the TA and student can converse about particular sections of the code. This arrangement simplifies the coreferencing task for both participants. The redesigned TA-tool also simplifies the interface by reducing screen clutter e.g. extraneous chat areas have been removed.

Acknowledgment

We would like to thank all the members of the TA-Tool group who have worked on developing early prototypes (including Sheeri Kritzer and Seth Landsman) as well as the students in the two courses in which the TA-Tool debuted.

References

[1] L. Brothers, V Sembugamoorthy, M Muller, ICICLE: Groupware For Code Inspection CSCW90 Proceedings, 1990.

[2] T. Richardson, Q. Stafford-Fraser, K. Wood, A. Hooper, IEEE Internet Computing, Vol 2. No 1, 1998.

[3] C.A. Ellis, S.J. Gibbs, and G.L. Rein, Groupware: Some issues and experiences *Communications of the ACM* January 1991/Vol.34, No. 1

[4] Clark, H.H., and Brennan, S.E. (1991). Grounding in Communication. *From Perspectives on Socially Shared Cognition*, edited by L.B. Resnick, R.M. Levine, and S.D. Teasley.

[5] Stefik, M., Bobrow, D.G., Foster, G., Lanning, S., and Tatar, D. (1987) WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Transactions on Office Information Systems*, 5(2), pp. 147-167.

[6] Landsman, S. and Alterman, R. (2001). Analyzing Groupware Usage in Context. Technical Report #CS-01-218, Computer Science Department, Brandeis University.