Extending the Personal Response System

Michael Gebauer
mgebauer@mit.edu

Advisor Professor Michael Ernst

Thursday, March 9, 2006
6.UAP, Final Report

1 Requirements

1.1 Overview

Over the past four years, various departments and groups at the Massachusetts Institute of Technology have invested in a hardware and software package called InterWrite PRS. PRS, which stands for the Personal Response System, is a way of dynamically polling groups for several purposes. For example, an instructor may use PRS to quickly and easily take attendance before a lecture. Additionally, an instructor may check comprehension during a lecture to understand how well students are learning. The system involves giving each student in a classroom a handheld infrared transmitter similar to a remote control. During a lecture and at the instructor's request students will press a button on the transmitter (this action is called "voting"). In real time the vote will be electronically registered and votes for all students can be tabulated and statistically analyzed by the manufacturer's software.

The purpose of this project is to extend the capabilities of PRS by writing new software to receive and tabulate votes. More specifically, this project dealt with one specific extension of PRS, which is to integrate PRS into the Groupthink Specification Exercise. The resulting software system is entitled "UpopVote".

The Groupthink Specification Exercise is a module in the UPOP (Undergraduate Practice Opportunities Program) IAP seminars. Briefly, a large group of students (up to 117 students) are arranged into teams of nine students each. Teams are given time to complete a partial specification of a telephone. Students are then asked several multiple choice questions which should be answered by the specification. Teams are awarded points based on number of agreeing answers.

The UpopVote project automates the process of tabulating these answers through PRS. The program displays the question and potential answers with a graphical user interface. A scoreboard and an indictor to inform participants when their answers have been recorded are also included in the user interface.

This report is intended to document the UpopVote software system. Because a significant challenge in developing this system was learning how to write software to interact with the PRS hardware, this documentation may also be useful for individuals seeking to write other software for PRS.

1.2 User Manual

1.2.1 System Requirements

The UpopVote system requires a Windows operating system. Additionally, the PRS SDK must be installed. Although an installer for this SDK is included with the UpopVote system, it is the property of GTCO CalComp, the manufacturer of InterWrite PRS. Consequently, users should only use the SDK for the UpopVote system. The manufacturer should be contacted about alternative uses.

The UpopVote system also requires EZ JCom to be installed. This is available from the EZ JCom website at the URL http://www.ezjcom.com/.

1.2.2 Defining Team Names

The current implementation of UpopVote supports exactly 13 teams. If less than 13 teams will be participating, the extra teams will maintain a score of zero. Modifying the Team enumeration defined in Team.java changes the number of teams playing.

When the game is initialized, team names are read from a text file. To locate this text file, first locate the top-level directory of the UpopVote program. By default this is named UpopVote. Within this directory, locate another directory named “upopvote”. This directory contains the text file, which is named “TeamNames.txt”.

Modify this file to rename the teams. Exactly the first 13 lines of this file will be parsed. Each string will be assigned to a team corresponding to the line number. For example, the string on line 6 will become name of team 6.

The folder “upopvote” and the text file “TeamNames.txt” are hard coded into the build and consequently should not be renamed unless the source code is modified.

1.2.3 Defining the Questions

The Groupthink Specification Exercise is divided into two rounds of questions. Similarly, the UpopVote program reads questions from two unique files. These files are located in the “upopvote” directory. This directory can be found in the top-level directory of the UpopVote system. Round one questions are read from the file named “round1question.xml”. Round two questions are read from the file named “round2questions.xml”.

To change the questions being used in the game, simply modify these XML files. Be careful not to modify the file names or paths, as these are hard coded into the UpopVote source code.

As the file extensions suggest, the question files are in an XML format. The file must begin with the "<upopvote>" tag and must end with the "</upopvote> tag. Within these tags, any number of questions may be encoded.

Each question must begin with the "<question>" tag and must end with the "</question>" tag. Between these tags, the question text should be typed without any tags. Following the question text, the answers should be typed. Answers will be displayed in the order they are entered in the XML file. Each answer must be enclosed by one of three supported answer tags.

- Permissible answers should be enclosed by the "<answer>" and "</answer>" tags.
- Strange yet permissible answers should be enclosed by the "<strange-answer>" and "</strange-answer>" tags.
- Impermissible answers should be enclosed by the "<wrong-answer>" and "</wrong-answer>" tags. Note that no points can be earned for wrong answers.

The following code snippet is an example of a question file containing answers, strange answers, and wrong answers.

```
<upopvote>

	<question>
		While the user is reviewing the answering machine messages, the
		phone company sends a ring signal.  Next the user presses talk.
		What happens?
		<answer>Nothing</answer>
		<answer>Both the user and caller hear the messages being
		reviewed</answer>
		<answer>The user and caller are connected; the answering machine
		is stopped</answer>
	</question>

	<question>
		The phone rings, and after 2 rings, the answering machine picks
		up.  The caller begins to record a message.  While the caller is
		speaking, the user presses talk.  What happens?
		<strange-answer>The answering machine continues to record; the
		user cannot speak to the caller</strange-answer>
		<answer>The answering machine continues to record; the user can
		speak to the caller</answer>
		<wrong-answer>The answering machine stops recording; the user
		cannot speak to the caller</wrong-answer>
```

```
            <answer>The answering machine stops recording; the user can speak
            to the caller</answer>
      </question>

</upopvote>
```

The UpopVote distribution comes with a set of sample questions. Review these files for a more thorough understanding of how to create new questions. Unfortunately, the parser for the question files is not particularly robust. Consequently, users should make sure that the question files meet the specification described here. Typos or unrecognized tags may cause the UpopVote program to fail.

1.2.4 Running the Application

When the UpopVote project is built, all compiled files and necessary dependencies are automatically arranged into an acceptable folder hierarchy. From the top-level directory there are three ways to run the program:

- To run the program and use PRS hardware type “java gui.UpopVote true” from the command prompt.
- To run the program without the PRS hardware (for testing only) type “java gui.UpopVote false” or just “java gui.UpopVote” from the command prompt.
- To run the program without the PRS hardware and with a thread that generates random votes type “java gui.UpopVoteTest” from the command prompt.

As an added convenience, new builds of the UpopVote system include three windows batch files that automatically execute the above options. These files can only be executed from Windows.

1.2.5 Player Registration

When running UpopVote, the game begins in a paused state. From this initial paused state, the user may enter the game’s registration mode. Clicking either the “Advance” button or the “Pause” button may enter this mode. Once registration mode has been entered, votes may be received and recorded from PRS transmitters. When a vote is received, the ID of the transmitter sending the vote will be displayed on the scoreboard. Next to the ID, the number of the vote will be displayed.

Each participant must enter some vote during registration mode to participate in the game. The software will not acknowledge transmitters that are not registered after exiting registration mode. It is not possible to register after exiting registration mode. It is never possible to un-register a player without restarting the software. Consequently, it is important that exactly each participant register for the game.

To exit registration mode and advance to round one, click the “Advance” button.

1.2.6 Round One Questions

After exiting the registration mode, the game immediately transitions into the first question of round one.

There are three general parts of the user interface: the question panel, timer, and scoreboard. The question panel is the large white panel in the upper left of the user interface. On this panel, all questions and answers are displayed.

The timer is the smaller panel in the upper right of the user interface. This panel includes a timer indicating how much time remains for the question. By default 60 seconds are allowed for each question. There are three buttons on the timer including the “Advance”, “Pause”, and “Add time” buttons. The “Advance” button un-pauses the game if currently paused, advances the remaining time to zero, or advances to the next question. The “Pause” button either pauses or un-pauses the game. Votes are not received if the game is paused. If the time is greater than zero, then the “Add time” button resets the remaining time to 60 seconds.

The scoreboard is the multi-colored table on the bottom of the user interface. The scoreboard serves many purposes. Player ID numbers are displayed on the scoreboard until a player has voted. When a team finishes voting, the number of points earned is displayed. When all teams finish voting, a summary of the votes received is displayed.

On the right half of the scoreboard, the teams are ranked according to total score. This team score ranking is dynamically updated as any team earns pointes.

1.2.7 Transitioning Between Rounds

Between rounds, a blank screen will be displayed. Clicking on the “Advance” button will reveal the most-improved scoreboard. Clicking again on the “Advance” button will advance to round two, immediately beginning the first question of that round.

1.2.8 Round Two Questions

Round two is very similar to round one. Two noteworthy changes are the addition of a most-improved scoreboard and a support for wrong and strange answers.

The most-improved scoreboard functions similarly to the total-score scoreboard. The score displayed on the most-improved scoreboard is the ratio of points earned in the second round to points earned thus far in the second round. Like, the total-score scoreboard teams are sorted from best to worst automatically.

Wrong and strange answers are supported in the second round. Wrong answers that somehow violate requirements of the system. Consequently no points are earned for this answer. Strange answers are somehow unusual but points will still be awarded for this answer. The bonus will not be awarded when all members of a team agree upon a strange answer. During voting answers, strange answers, and wrong answers are all displayed in black. After voting has finished, answers are displayed in black, strange answers are displayed in blue, and wrong answers are displayed in red. In the vote summary section of the scoreboard, wrong answers are again displayed in red and with an asterisk.

Once the last question has been completed, the game finishes and the user interface cannot be manipulated in any way.

1.2.9 Game Summary

When the user interface closed, a summary of the game results will be written to a text file. The text file is named summary.txt and is located in the “upopvote” directory of the top-level directory of this software system. This text file will be written regardless of whether the game has been completed or not. The text file is overwritten the next time the UpopVote program is closed.

This format of the text file consists of the question (encoded in html) followed by a matrix of votes received for each player. Each row of the matrix represents a team, while each column of the matrix represents a player on the team.

1.3 Requirements

UpopVote was written primarily in Java and thus requires version 5.0 or newer of the J2SE Runtime Environment (JRE). To build the UpopVote project, version 5.0 or newer of the J2SE Development Kit (JDK) is required.

UpopVote's interaction with the PRS hardware is abstracted away by an ActiveX Control and a software package called EZ JCom. GTCO CalComp, the InterWrite PRS manufacturer, has made a SDK based on an ActiveX Control available, which enables programmatic interaction with the PRS hardware. Although this tool is available for the UpopVote system, it is property of GTCO CalComp. Consequently, the company must be contacted for permission of this tool is to be used for purposes other than UpopVote. The SDK should be installed from the manufacturer-supplied installer. Because this SDK is only available for Windows operating systems, the UpopVote project is currently limited to Windows as well.

A tool called EZ JCom provides the bridge between the ActiveX Control and Java. EZ JCom is a tool designed specifically for this purpose. Several alternative tools are available for bridging ActiveX Controls and Java. It should be possible to use any of these alternatives with a bit of reworking of the UpopVote source code. This is more thoroughly described later in this documentation.

The builds of the UpopVote system require approximately one megabyte of disk space.

2 Design

The UpopVote system is primary written in Java, but depends on some non-Java elements to play critical roles. Particularly, GTCO CalComp supplied a "software development kit" to interface between the PRS hardware and other programs. Another tool called EZ JCom was needed to bridge between the software development kit and Java. The remainder of the project is Java.

Excluding packages used only for tests, UpopVote is divided into four packages. The packaged named "ezjcomGenerated" represents classes automatically generated by the EZ JCom tool. The "simpledomparser" package represents the XML parser taken from an online tutorial [Yang 2002]. The majority of Java classes written for UpopVote are for the user interface. These classes are all contained in the "gui" package. The final package, entitled "scorekeeping", contains all classes that are intended to store question and vote information.

2.1.1 The PRS Software Development Kit

The fundamental goal of the UpopVote project is to accept information from the PRS transmitters and then to tabulate and convey this information to the users. Two tools are used to bridge the gap from the hardware to Java. These two tools are the PRS software development kit and the EZ JCom tool. The PRS SDK is somewhat of a black box. It is installed from a Windows executable file and is minimally documented. This section is an attempt to explain all that has been learned about this black box.

The installer creates a small application that is used to test PRS hardware. The application is called “Test PrsX”. It is a small user interface which prints information for any votes received from PRS transmitters. This application is not useful for writing PRS software.

Only through experimentation has it become somewhat apparent how the development kit works. The fundamental part of the SDK is an ActiveX Control named PrsX.ocx. By default, this file is installed into the C:\WINDOWS\system32 directory on Windows computers. An ActiveX Control is a type of component for Windows machines. Although ActiveX Controls can be written in various languages, this particular one seems to be written in Visual Basic (although the language is irrelevant). The ActiveX Control makes a number of operations or methods available, including those which are necessary to programmatically access get data from PRS transmitters.

It should be stated that if any individuals working on this project in the future have a more thorough understanding of Visual Basic or ActiveX Controls, they might be able to clarify this documentation and the structure of the software development kit.

2.1.2 EZ JCom and the ezjcomGenerated Package

EZ JCom is an important part of the system. From a high level, the EZ JCom tool and associated classes in the “ezjcomGenerated” package should be thought of as the bridge between the manufacturer-supplied PRS software development kit and the Java classes that are the bulk of the UpopVote project.

EZ JCom is another Windows based program which most be downloaded and installed from the manufacturers website [EZ JCom 2005]. The installer installs several libraries and obscure files, which are necessary for UpopVote to function correctly. This includes two files named EZJcomLib17.dll and EZJcomMTALib.dll, which are installed into the

C:\WINDOWS\system32 directory by default. The installer does this automatically. Two additional files named EZJcomLib17.lib and Jprsx.dll must be included in the top-level directory of UpopVote. This is done automatically when UpopVote is built using the included build.xml file.

Running the installed application walks the user through the steps necessary to create the bridge. The user is asked to input the ActiveX Control and answer a few specific questions about the computer being used. The application outputs a package with Java classes. The Java classes include methods that enable votes to be received from the hardware. These generated files can be found in the ezjcomGenerated package.

There are three generated Java classes named ctlPrsX, _ctlPrsX, and __ctlPrsX. The later is an abstract class that must be implemented manually. The abstract method of this class is named Receive. This method is executed when a vote is received from a PRS transmitter. In UpopVote, the scorekeeping.EZJComUpopVoteConnector class is the implementation of the __ctlPrsX class.

To see how to instantiate objects from these classes, refer to the UpopVote class.

2.1.3 The scorekeeping Package

The scorekeeping package contains all classes written specifically for the UpopVote project but not a part of the user interface. These classes are documented. Developers should refer to the comments to gain an understanding of how the classes are used.

2.1.4 The gui Package

The used interface for UpopVote is written in Swing. Again, developers should refer to comments in the classes to gain an understanding of how the classes are used.

2.1.5 The Parser

An XML parser is used to read question files. This parser is contained in the simpledomparser package. The parser was taken from an online tutorial and reused [Yang 2002].

3 Testing

3.1 Strategy

Following the first implementation of the UpopVote system, unit test were written for all abstract data types in the “scorekeeping” package.

Unit tests were not written for the “ezjcomGenerated” package because the EZ JCom software automatically generated all classes contained in that directory. It is assumed that the manufacturer has tested the package. This assumption seems to be supported by black box tests of the complete system.

Similarly, unit tests were not written for the “simpledomparser” package because all classes in that directory were copied from an online XML parsing tutorial. This reference is documented in the bibliography.

Unit tests were not written for the “gui” package. This should be completed in the future.

Black box testing for the system was conducted manually. A class entitled UpopVoteTest contained the entry point for black box testing of the UpopVote system. Instantiating the class automatically instantiates the UpopVote user interface. Additionally a thread generates random votes at random intervals. This enabled much of the UpopVote code to be tested. The tester would manually manipulate the user interface to ensure proper functionality.

One shortcoming of this style of black box testing is several parts of the system were not included in the tests. Specifically, neither the ezjcomGenerated classes nor the PRS SDK were employed in these black box tests. The complete system was tested manually, but less frequently that would be ideal. In the future, it might be useful to develop a better way to simulate votes for black box testing.

3.2 Test Results

During development, testing was not conducted as rigorously as it should have been. Consequently, there is a burden to implement necessary tests and documentation after the first version of the software has been implemented. This suggests limited confidence in the testing.

It should be noted, that the complete system has been successfully used with a large group. This suggests some level of confidence with the software.

4 Reflection

4.1 Evaluation

During the second UPOP IAP Session in January 2006, the UpopVote software was successfully used with a large group of approximately 115 participants. The participants seemed to have genuinely enjoyed the Groupthink Specification Exercise. During the contest several teams cheered when they earned bonuses, groaned when they came close, and applauded after the activity. It seems that the UpopVote software positively impacted the exercise. It is unfortunate that a software bug prevented a similar outcome during the first UPOP IAP Session.

The focus now becomes to ensure that the software can be reused and improved in future years. This will require much documentation and testing. Unfortunately, this was not done as thoroughly as it should have been during initial development of the software. If this work can be completed, the project will have been successfully completed.

4.2 Lessons

The UpopVote software was first used on a large scale during the first UPOP IAP Session in January 2006. A bug in the VoteReceiver class prevented ID numbers from being removed from the user interface when a vote was received. As a consequence, participants were unsure of whether their votes had been recorded and were forced to keep voting repeatedly to ensure their votes had in fact been recorded. This bug substantially increased the number of votes being received by the system and made it difficult for everyone to get at least one vote recorded.

The black box testing of the complete system was being done manually. That is the software was run and the PRS transmitters were being physically used to send votes to the software. This approach had been chosen because it employed the complete system including the PRS SDK.

Ultimately this style of testing failed to identify the bug that kept votes from being removed, because not all tests were conducted. If the tests had been somehow automated, it

would have been possible to ensure that all necessary tests were conducted after each update to the code. This failure is a testament to importance of writing a complete set of test cases.

4.3 Known Bugs and Limitations

At least one bug remains in the UpopVote project. When running the software with a large number of participants, the following exception is often thrown:

Exception during EZJCom Event processing in scorekeeping.EZJComUpopVoteConnectorReceive: java.lang.RuntimeException: Cannot find ID: 0 on any team.

at gui.Team.findTeam (Team.java:138)
at scorekeeping.VoteReceiver.receive (VoteReceiver.java:96)
at scorekeeping.EZJComUpopVoteConnector.Receive (EZJComUpopVoteConnector java:42)

The exception suggests that a vote is being received from transmitter with ID equal to zero. This is not a valid ID. Thus far, this except has not been recreated in a testing setting.

It has been suggested that this exception occurs when two votes are received from two different PRS transmitters at the same time. There could be some sort of collision that causes neither vote to be sent to the UpopVote code. Markos Hankin, who used to be responsible for maintaining the PRS in some classrooms at MIT, found this possibility to be credible.

Although there are not any other known bugs, several limitations do exist. In particular, the following limitations may be good improvements to include in a second version of the UpopVote system.

- It should be possible to resume an incomplete game by reloading questions and scores earned in a new instance of the program.
- The parseQuestionsFile(File f) method of the UpopVote class is not robust. It fails when an unsupported tag is encountered. It would make sense to make this more effective.
- It is not possible to add or remove players when a game is in progress. This would be a nice feature.

5 Appendix

5.1 Logistics

For questions concerning the PRS hardware, it is useful to contact the manufacturer GTCO CalComp. The manufacturer's website is http://www.gtcocalcomp.com/. Several phone numbers, email addresses, and online forums are available for further assistance from the website.

For questions concerning the equipment in either TEAL room at MIT (currently this includes 26-152 and 32-082), it is useful to contact the TEAL Program Coordinator. Currently this is Andrew Neely. The TEAL rooms should be reserved through the MIT Schedules Office.

5.2 Troubleshooting

An interesting nuance of PRS is the customer ID. The customer ID should not be confused with the transmitter ID that is printed on the back of each transmitter – they are different. The customer ID is programmed into the firmware of each transmitter. The customer ID is not well documented, but is intended to enable customers to order PRS transmitters with unique transmitter IDs. GTCO CalComp should be contacted for more information about this.

At MIT, in 26-152 there are PRS transmitters programmed with two different customer ID numbers. When the PRS SDK is installed, some PRS transmitters are not recognized as a consequence. To remedy the problem, access a file installed by the SDK in the C:\WINDOWS directory called prs.ini. In this file, change the line stating "id=5y?YGCCF" to "id=4r8ZNCCF". If the new customer ID does not fix the problem, contact GTCO CalComp and explain the issue. They should be able to determine a new customer ID that will work for all transmitters.

6 Works Cited

EZ JCom. Desiderata Software. 10 Dec. 2005 <http://www.ezjcom.com/>.

GTCO CalComp. 15 Nov. 2005 <http://www.gtcocalcomp.com/>.

Neely, Andrew. Personal interview. 6 Dec. 2005 – 31 Jan. 2006.

Pfeffer, Mike. mpfeffer@gtcocalcomp.com. "PRS SDK..." Email to the author. 16 Nov. 2005 – 17 Jan. 2006.

Yang, Guang. "Build Your Own Lightweight XML DOM Parser." DevX. 22 Nov. 2002. Jupitermedia Corporation. 1 Dec. 2005 <http://www.devx.com/xml/Article/10114#codeitemarea>.