Developing an Improved, Web-Based Classroom Response System with Web Services

Senior Honors Thesis By:

Oleg B. Seletsky
Oleg.B.Seletsky.09@Alum.Dartmouth.ORG

Advisor:

Professor Sean W. Smith
Sean.W.Smith@Dartmouth.EDU

Dartmouth Computer Science Technical Report TR2009-651

June 03, 2009
# Abstract

3

# 1 Introduction

3

# 2 My Experiences with CRS at Dartmouth as an Undergraduate

## 2.1 A Basic, Home-grown CRS

6

## 2.2 The more High-Tech Commercial CRS

8

# 3 The Design and Functionality of Our CRS

## 3.1 Motivation

9

## 3.2 The User Interface

### 3.2.1 Student User Interfaces

10

### 3.2.2 Instructor User Interfaces

12

## 3.3 The Internals of Our CRS

### 3.3.1 Web Services

16

### 3.3.2 The Database Schema

19

### 3.3.3 Authentication

20

### 3.3.4 Confusion Meter

21

# 4 A Comparison of CRS Types Commercially Available Products

## 4.1 CRS Types

22

## 4.2 Commercial CRS Products

### 4.2.1 Numina II SRS

24

### 4.2.2 Poll Everywhere

24

# 5 A Clicker Study at a University of Wisconsin

## 5.1 Exciting Results

25

## 5.2 Problems that Can Arise

26

# 6 Evaluation of Our System

27

# 7 Future Work

## 7.1 Development

29

## 7.2 Testing

30

# 8 Conclusion

31

# Acknowledgements

32

# Code Availability

32

# Works Cited

33

# Appendix A: Descriptions of Implemented Web Services

34
Abstract

Classroom Response Systems (CRS) are an in-class technology used to poll students and instantly display an aggregate representation of their responses. CRS have been around since the 1970s and have become increasingly more popular in higher education lecture halls. Even though technology, specifically computers and communications, has improved significantly since the 1970s, CRS have remained surprisingly unchanged. The purpose of this project was to develop an innovative web-based CRS using web services. The web-based aspect utilizes Dartmouth's wireless campus while the web services back-end makes the product more extensible. Lastly, we added a set of out-of-class learning tools for students as well as an in-class tool called the Confusion Meter to enhance student-to-instructor communication. With these features, our goal was to create a free, open-source system that enhances the teaching and learning experience and remains extensible and developer-friendly, unlike any commercial CRS currently available.

1 Introduction

The project was a senior thesis under Professor Sean W. Smith of the Computer Science Department at Dartmouth College. Professor Smith first suggested the idea of creating a web-based CRS to eliminate the need to purchase extra hardware and because web browsers can provide more flexibility than a numeric keypad. After meeting with members of Dartmouth’s Kiewit Academic Computing, we decided to explore the use of web services as a back-end to our web-based system. Mark O’Neil of Academic
Computing was working on a set of web services that exposed Blackboard\textsuperscript{1} functionality such as authentication and enrollment. We decided to implement the server-side of our system using web services to increase its extensibility and possibly interface with Blackboard. Professor Smith used the CRS in his Computer Architecture course (CS37) during the second half of the Spring term to test the functionality and student reception.

I will begin with an overview with the basic components of a CRS followed by a brief history of the technology. I will then discuss my experience with two CRS as an undergraduate at Dartmouth. The first is a simple, home-grown CRS used in an introductory physics course and the second is the popular TurningTechnologies TurningPoint system used in an introductory biology course. After this I will explain the design and implementation of our system. I will discuss the various user interfaces it provides as well as the back-end web services and database schema. I will also focus on the combination of features that help make our system unique and better than other systems out there. I will then discuss the different types of CRS as well as compare two of the commercially successfully systems out there. I will conclude by discussing the successes and failures of our system in CS37. This will include the results of a student survey as well as input from Professor Smith. Lastly, I will discuss where we hope this project will go in the future, including the features we would like to see added and what further testing we would like to perform in classroom environments.

\textsuperscript{1} Blackboard, \url{http://www.blackboard.com/}, is the course management system used at Dartmouth. Professors often use it to post documents and grades.
The main components of a classroom response system (CRS) are the response input device, the computer that handles aggregating the responses, and the display. The professor will ask a question, wait for the students to respond using their input devices, and then display the distribution of responses. A discussion will usually follow if a significant number of students responded incorrectly. There are variations on functionality, input device type, and output type but this is the standard order of events. The main goals of CRS are to encourage class participation, allow the professor to gauge how well the class is following along, and increase student comprehension and assessment scores.

The original CRS were often built into lecture halls, with each seat having its own input device wired to the central computer. Even the earliest systems had both anonymous and non-anonymous modes to allow the professor to record only the overall response distribution or every individual's response. It is important to note that student responses, even in these earlier systems, were confidential between students. Overall, CRS had a slow start in higher education. As Judson and Sawada note, "...research from the 1960s and 1970s related to electronic response systems does not support the claim that student achievement will increase as a result of using such systems (Judson and Sawada 2002, 172).” Nonetheless CRS did not die out. This may be because "students provided overwhelming endorsement for the electronic response systems [CRS]...(Judson and Sawada 2002, 173)" Another reason could be improvements in technology have allowed the CRS to become more mobile. These days it is rare to use any electronics that have to be physically attached to a network. Whether using a TV remote or sitting on the College green and browsing the Internet, people have gotten very accustomed to taking
electronics anywhere and using them without *plugging in*. Wireless technology has been applied to CRS with the development of Infrared (IR) and Radio Frequency (RF) keypads. These wireless keypads typically communicate with a receiver attached to the professor's computer. The supplied software then tallies up the responses and displays the distribution. The IR keypads are more primitive, allowing only one-way communication between the keypad and the receiver. In contrast, RF keypads allow communication in both directions thereby allowing the student to know if he or she answered the question correctly. The IR and RF keypads came around in the 1980s and make up the 2nd generation of CRS. The 3rd and current generation of CRS is web-based systems (Lowery 2005, 2). These systems have become more popular as college campuses have become wireless. Here at Dartmouth, over 99% of students have a laptop and our wireless campus provides the perfect environment for a web-based CRS.

2 My Experiences with CRS at Dartmouth as an Undergraduate

2.1 A Basic, Home-grown CRS

I used the first system in my introductory physics course in Fall 2005. The system was very primitive yet seemingly effective just like the first CRS used in the 1970s. Upon entering each class period, each student picks up a key ring with a set of colored squares and corresponding numbers (Figure 1). Throughout the lecture, the professor displays multiple choice questions and asks the students to respond by holding up the card with an answer choice. The professor then says which choices were the most common and picks a student from each ‘response group’ to explain why they selected

---

1 All images and diagrams in this paper were produced by the author.
that choice. After the professor reveals the correct response, a classroom discussion follows. This system is perhaps even more basic then the earliest systems of the 1970s described previously since it relies on very limited technology. However, one might make the claim that it is wireless! Nevertheless it was certainly successful in generating class discussion and provided the professor with information on how well the students understood the topic. Unfortunately it lacked the student-student response confidentiality present in some of the earliest CRS of the 1970s.

**Figure 1:** Cards used in introductory physics

![Figure 1](image)

**Figure 2:** Clicker used in introductory biology

![Figure 2](image)
2.2 The more High-Tech Commercial CRS

The second CRS I used was TurningTechnologies’ TurningPoint system in my introductory biology class in Fall 2006. The system is based on RF clicker technology (Figure 2). Students were encouraged to purchase the clickers, which were available at Dartmouth’s Computer Store. The clickers could only be purchased using a student’s college ID card, thereby linking every clicker to the student who purchased it. The professor embedded six to eight questions into the PowerPoint presentation for each hour long lecture. In lecture, the professor clicks a button to open a multiple-choice question for polling. Students then press a number on the clicker to submit their response. The professor can see the number of students that have responded and once this number is high enough, the professor closes the question and a graph showing the distribution of responses is displayed. The professor could later access all of the students’ responses for grading purposes. However, because the program was a pilot, the clickers were used only in anonymous mode and a student’s participation or responses did not impact their course grade. From a student’s point of view, this system did not have many more features compared to the CRS used in the physics course even though it clearly relied on much more advanced technology. Perhaps the most important feature for the students was the student-student response privacy that the clickers allowed.

Both systems were similar in that neither provided out-of-class functionality that we developed in this project. Our biology professors posted online the lectures which contained the questions. However, the question slides were often missing the distribution or the correct response wasn’t identified. They also didn’t provide students with their personal responses during the lecture. It was up to the student to take notes about the
question and their response. Interestingly Judson and Sawada cite Littauer, a professor who ",...attempted to lessen the information load on students by distribution sheets before each lecture listing the question and answer choices. In this way he liberated students from note taking. (Judson and Sawada 2002, 171)”. This is an interesting solution to a problem that seems to be present in every CRS out there, including the two that I have used at Dartmouth within the past four years. Our system implements a different solution for this issue by providing students with out-of-class features that eliminate the need to take notes about questions, the class distribution, the correct response, or the students’ personal response.

3 The Design and Functionality of Our CRS

3.1 Motivation

After surveying the commercially available CRS, we saw the need to develop a new system. We sought to add functionality that would go beyond the standard ask-respond tool found in all CRS, so we created a set of tools that would let students continue learning from the questions outside of the classroom. In addition, we wanted our system to remain open-source and extensible. Since there are many potential input devices (laptops, PDAs, smartphones, etc.), we built our server-side using web services (WS) allowing easy creation of new client applications. The benefits of using WS and how we implemented them is discussed further in Section 3.3.1. We also sought to create a CRS that did not need extra hardware such as clickers. Thus, we made the client-side of our system web-based to take advantage of Dartmouth’s wireless campus and that 99% of the student body has a laptop.
3.2 The User Interface

The user interface (UI) in our system is completely web-based. Our system has separate user interfaces (UIs) for in class and out of class use, as well as for students and instructors. This yields a total of four different UIs. Our goal was to identify the functionality needed by each user role in each setting and make this functionality easily accessible while maintaining a simple, easy-to-use UI.

3.2.1 Student User Interfaces

The IN CLASS STUDENT UI web page contains three frames (Figure 3). The upper left frame contains an HTML form used to set the Confusion Value and the upper right frame contains an "Answer Current Question" hyperlink. Once the professor opens a question for polling, the student clicks this link and the question loads into the main frame of the page. The student can submit responses until the professor closes the question; the last response is recorded.

Figure 3: A student responding to a question.

your previous answer to this question was 1

Question 266: Capital of Australia

What is the capital of Australia?

- 1: Canberra
- 2: Sydney
- 3: Melbourne

Submit Choice
The OUT OF CLASS STUDENT UI contains a set of tools for students to use outside of class (Figure 4). The most basic feature allows a student to click on any question that they responded to during the lecture and see their response and the class distribution. On this page, there is also a discussion thread associated with the question. We thought students might come up with questions or need further explanation outside of class. To remedy this, the system provides a simple, forum-based collaboration environment where students and the instructor can post questions and comments. Each question has a unique Question Id which can be used by students and the instructor to reference questions directly. The upper left frame of the OUT OF CLASS UI contains a Search by Question Id form to quickly access questions. The Question Name and Question Text attributes of each question are searchable making it easier to find questions about specific topics. A student can easily create mini quizzes by checking off which
questions they would like to appear in the quiz and then clicking the Make Quiz button in the left frame. The system will fetch the list of questions and once the student chooses a response for each, the system will return a score and the list of graded questions. All of these features are intended to extend the learning experience beyond the typical and limited ask-respond-discuss, in-class process of other CRS.

3.2.2 Instructor User Interfaces

The IN CLASS INSTRUCTOR UI contains three frames as well (Figure 5). The upper left frame displays the aggregate results of the Confusion Meter. The upper right frame contains links to create Lectures and Questions. The lower left frame contains a drop down list of lectures defaulting to the most recent one, enabling recently created questions to be more accessible. The professor can select a Lecture to fetch a list of Questions in that Lecture and then click on a question to fetch it into the main frame. From here the professor opens the question and sees a real-time count of the number of responses. The professor then closes the question and a distribution of student responses is displayed.
**Figure 5:** An instructor preparing to ask a question in class.

**Figure 6:** An instructor preparing to create a new question.

**Question 266: Capital of Australia**

What is the capital of Australia?

1: Canberra  
2: Sydney  
3: Melbourne
The INSTRUCTOR’s OUT OF CLASS UI provides more tools to work with questions (Figure 6). In the lower left frame, the professor can search the Question Name and Question Text just like the students. For each Question, the professor has the option to ask, view, and edit the Question. The ask link has the same action as asking a Question in the IN CLASS UI. The view link allows the professor to view the distribution of the question responses and the discussion thread (mentioned previously). The professor can also archive and unarchive questions. This is equivalent to deleting a Question except a professor can always recover the question and all the data stored for it by unarchiving it. Students cannot view Questions that have been archived. Lastly, the professor can edit any attribute of the Question by clicking on the edit link.\(^1\) The upper right frame provides a link to Create a Question and Create a Lecture. Each Question must be associated with a Lecture. This makes it easier for the professor to quickly find questions to ask in class. Each Question has a Question Name, Question Text, up to 10 Question Choices, up to 2 Images\(^2\), Correct Choice Id, and Lecture. There are two kinds of questions, IN CLASS and OUT OF CLASS. Originally, a student could only view a Question in the OUT OF CLASS UI if the student had responded to that Question during a previous lecture. However, Professor Smith suggested students may want to see practice questions before an exam. Thus, we created an OUT OF CLASS question that a student can see without having responded to it previously. Students can search, discuss, and make mini quizzes out of these questions as well.

\(^1\) Currently a professor cannot change the images that were uploaded during Question creation.
\(^2\) The limits on number of choices and images are only on the client side.
The *Confusion Meter* is a tool that Professor Smith designed. He wanted to know if the pace of his lecturing was too slow or too fast without continuously asking questions. He was looking for a passive tool that would allow real-time student-to-instructor communication. Interestingly, Garg describes similar functionality in a CRS he was using in the 1970s: “The system has an update feature whereby the voting and display are set up in a continuous mode. Instructor can observe the feedback display and modify his presentation in response to student reaction regarding pace such as "go faster", "go slower," "about right", or dealing with any other attribute such as quality of presentation, for example, "excellent", "good", "fair", "poor", "terrible"(Garg 1975,6)”

Our original version allowed each student to set a *Confusion Value* of 1-10 to specify how well or poorly they were following along. The professor would then see an aggregate sum representing the entire class' *Confusion Value*. However, we decided that this feedback did not provide a good reading of the distribution of *Confusion Values* among the students. The improved version allows students to set their confusion value as "Know this Stuff/Bored," "Good Review," "Okay," "Confused," or "Very Confused" (Figure 7). The system displays a histogram for professor showing the distribution of the students’ *Confusion Values* (Figure 8). The histogram refreshes on a set time interval to allow close to real-time feedback.

**Figure 7:** The *Confusion Values* a student can choose from.

<table>
<thead>
<tr>
<th>KNOW THIS STUFF/BORED</th>
<th>GOOD REVIEW</th>
<th>OKAY</th>
<th>CONFUSED</th>
<th>VERY CONFUSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>update confusion value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15
3.3 The Internals of Our CRS

Our system is a web-based, client-server application developed using mainly PHP and MySQL and web services. I will begin by explaining web services and how we implemented them in our system. I will then provide an overview of our database schema. Lastly I will focus on how we implemented a few of the components.

3.3.1 Web Services

Web services (WS) are a method to create Internet application programming interfaces (APIs). They allow a client to send a request to a remote server to perform a function and then send the result back in a response. Both the request and response are usually XML and the input and output message formats of the web service are usually defined by the web service’s WS definition language (WSDL) file. The messages are transported over HTTP. WS allow one to call functions in framework and system independent ways. For example, an application written in Java on an iPhone can call WS written in PHP and hosted on a Linux server because the communication is defined by a language independent XML Schema.
Figure 9: Application without WS

Figure 10: Application with WS
WS allow us to create a level of abstraction between the client and the server. The job of the client is reduced to rendering the UI and calling services. The two diagrams above (Figures 9 and 10) compare what our system design would have been without WS and what it actually is. The most important part to notice in the first diagram is the presence of the pink ‘Backend Application Logic’ throughout all the clients. This means that every client that we want to create for our system will have to interact directly with the database and handle all the application logic involved in logging in, creating questions, responding to questions, etc. However, by taking advantage of WS, we ‘concentrate’ all the application logic in one place and allow all clients to use it through WS. These WS interface with our database to execute system logic. If we wanted to create another client for our system, say an iPhone application, we wouldn’t have to re-code all of the backend application logic. The iPhone application would simply call the same services. In fact, any development platform that supports WS, most of which do, can take advantage of our set of services. Similarly, to change the authentication method, one would only need to change the authentication WS and all clients would start using the new authentication system (assuming the API remains the same). Using WS also allows our system to interface with other software systems that expose their functionality as WS.

In our system, the web services were implemented using NuSOAP, a PHP library that allows creation and calling of web services. Many development platforms support WS, but NuSOAP was chosen because it is very lightweight, uses PHP which we used for our front-end, and because it is open source like the rest of our system. NuSOAP is a set of classes, and not a PHP extension, meaning that ‘installation’ only requires copying
a set of PHP files. This makes it trivial to move our server-side code between servers.

Unfortunately, NuSOAP is still in relatively early stages of development (yet to reach version 1.0). NuSOAP documentation is hard to find on the Internet, so we mostly used tutorials and forums to understand its functionality. In retrospect it may have been a better idea use a more widely used and supported web services library (perhaps a Java one).\footnote{Throughout development it was sometimes difficult to fix bugs and/or determine if other users had run into the same problems. A more commonly used library would have made this process easier.}

### 3.3.2 The Database Schema

The schema for the MySQL database used by the system contains 10 tables (Figure 9). The main tables in the system are USERS, QUESTIONS, and COURSES. Both user roles, students and instructors, are mapped to the courses they are enrolled in (or teach) through the USERS_TO_COURSES table. Each question is mapped to the course it was created for through the `course_id` column in the QUESTIONS table. Each question can have any number of choices found in the QUESTION_CHOICES table, but the client currently only supports up to 10 choices. Each question can also have any number of media elements (images, movies clips, sound clips, etc.) found in the MEDIA table. However, the client currently only supports up to two images. The discussion thread associated with each question is stored in the POSTS table. QUESTION_RESPONSES table stores the user’s last response for each question. Users’ current confusion values for each course are stored in the CONFUSION_VALUES table.
3.3.3 Authentication

The system has its own authentication system (i.e. it does not rely on external systems such as Blackboard). The USERS table contains three columns that are used for authentication (username, password, and user_role). The password is stored in encrypted form and the user_role can take on two values: INSTRUCTOR and STUDENT. These roles are used to control which UI is displayed for a user as well as for authorization.
purposes within the web services. Upon successful authentication and course selection, the following three cookies are stored on the user’s machine:

- **PTC_COURSE**: stores the internal unique course id of the course the user selected upon logging in
- **PTC_ROLE**: stores the user role described above
- **PTC_TOKEN**: contains a hash of a combination of the user’s username and password
- **PTC_USER**: stores the internal unique user id of the user
- **PTC_GUI**: stores the UI selected by the user upon login (IN_CLASS or OUT_OF_CLASS)

Every time a web service is called, the set of values in PTC_COURSE, PTC_ROLE, PTC_TOKEN, and PTC_USER are validated. The PTC_TOKEN must match the string generated by running the hashing function on the *username* and *password* corresponding to the *user_id*. The user must also have the stored role and must be enrolled in the course with *course_id*. If any of these values have been changed by the user through cookie manipulation the authentication process would fail and a SOAP error will be returned.

### 3.3.4 Confusion Meter

The Confusion Meter graph, as well as the response distribution graphs, is generated using PHPGraphLib. A web service is called to retrieve the input values into the graph, but the graph itself is generated by the client-side. The graph can be set to

---

1 Currently the system allows a student to choose either when in class. Perhaps in the next version, the professor would set a flag at the beginning of each class session so students would be automatically redirected to the IN CLASS UI and not be able to access the OUT OF CLASS UI until the professor turned the flag off at the end of the class session.

2 In hindsight it would have been a better idea to store a simple encrypted string of a combination of all of these values and simply decrypt the string every time a web service is called and if it the string had been altered, the parts would not be extracted properly and the request would fail. This would significantly decrease the load on the system. This is a potential improvement for the next version.
refresh at any given time interval (currently set to refresh every minute). A student can set his or her confusion value at any time using the UI. The student’s Confusion Value for a given course is stored in the CONFUSION_VALUES table. The Confusion Value input page is set to refresh on a set time interval (currently set to five minutes). Upon refreshing, the page set the student’s confusion to ‘OK.’

4 A Comparison of CRS Types Commercially Available Products

4.1 CRS Types

As mentioned in the introduction, there are currently three main types of mobile CRS systems: Infrared (IR), Radio Frequency (RF), and web-based (or WiFi). The IR systems were developed first. The input devices in these systems are the least expensive compared to the other two. However, IR systems have problems that are inherent to the technology. IR technology only works with direct line-of-sight, meaning that within large lecture halls, it is often necessary to set up many receivers so students are able to aim the ‘clicker’ directly at the receivers. The communication between the input device and the receiver is also one-way meaning that students cannot receive any confirmation that their response was registered. The range is also limited to approximately 80 ft, rendering the system unusable in large lecture halls of over a couple hundred students.

RF input devices look very similar to IR ones. However, RF systems do not require line-of-sight for responses and allow two-way communication allowing students to receive

---

1 The graph had originally been set to refresh every five seconds but this was distracting to the students. A one minute lag should be close enough to real-time in this situation.
2 Note that this is only done on the client side. If a student sets a Confusion Value and then closes the web page, the student’s Confusion Value will not reset to ‘OK’ until the student loads the web page. Creating a database-side reset trigger may be worth pursuing in the next version, but since a student should be logged in at all times during a class session, this implementation shouldn’t pose serious problems.
confirmation that the response registered. RF systems also have a larger range, and are thus more suited for larger lecture halls. RF systems are more expensive than IR systems mostly due to ‘clicker’ prices. Lastly, web-based systems are the latest in CRS technology. As mentioned earlier, our system falls in this category. Web-based systems have gained much popularity as campuses have become wireless and more students own laptops\(^1\). These systems allow more complex input than standard keypads. Unfortunately one of the problems with web-based systems is the opportunity for off-task behavior (Lowery 2005, 13). However, a study by Ward et al. showed that while using the Numina II SRS, a web-based CRS described in the next section, at least 80% of the students were on task at any given time. Web surfing, e-mail, and solitaire were cited as the most common off-task activities (Ward et al. 2003,11).

### 4.2 Commercial CRS Products

There are a lot of commercially available CRS systems. They range in communication methods, input devices, pricing schemes, and the software that they interface with. Earlier I discussed my experiences with the commercial RF clicker-based TurningPoint system. I will discuss two more systems here. The first is a web-based system developed at a university with the goal of being used on PDAs. The second is an audience response system (ARS) that uses text messaging as its communication method and is geared towards higher education as well as business presentations and meetings.

---

\(^1\) Laptops can serve as the input devices. Purchasing Wi-Fi enabled input devices for a CRS is often cited as one of the main drawbacks.
4.2.1 Numina II SRS

The Numina Project was started at North Carolina Wilmington at the same time as the campus was beginning to deploy its first wireless network. The goal of the project was to develop tools for, and explore the benefits of using mobile technology within the classroom setting. The project has used a set of handheld devices through its time. The original ones cost as much as $750 but the newer and more powerful ones cost around $400 (Heath et. Al 2005, 47). The Numina II SRS is a completely web-based system like ours. Consequently, this makes the system hardware and operating system (OS) independent. Unlike our system which uses standard web pages for its front-end, the Numina II SRS is based on Macromedia Flash technology. This allows the system to be a lot more interactive and supports a larger set of question formats and responses. For example, the professor has the ability to create a question with an image, such as a graph, and ask students to point to a coordinate with a given property. There of course is no exactly right answer so the professor sees a distribution of where the students clicked right on the image of the graph itself. This sort of interaction would certainly be possible in a system like ours (using either Flash or Java Applets). However, at the time of the design we decided to stick with a simpler in-class input system and limit it to multiple-choice questions.

4.2.2 Poll Everywhere

Poll Everywhere is a CRS that uses text messaging as its basis of communication. This system takes advantage of the fact that almost all students have cell phones. Unlike with web-based communication at most universities, text message communication is not
free and the payment burden lies with the owner of the cell phone. Additionally, cell phone service may not be as available inside lecture halls as a campus-wide WiFi network. The system has the main in-class ask-respond features that CRS are ‘required’ to have. The product website, www.polleverywhere.com, advertises Poll Everywhere for businesses and non-profits as well as K-12 and Higher Education. Questions can also be answered through the Web by clicking on a link sent out by the presenter. The system also allows the results to be added to the aggregate graph as they come in. Poll Everywhere is free for up to 30 participants per poll.

5 A Clicker Study at a University of Wisconsin

As CRS become more popular, more and more studies on their impact on student grades, class participation, and overall satisfaction are being done. The University of Wisconsin study was a large scale assessment of a clicker-based CRS. 28 faculty members and 3,500 students were asked to complete an online survey after using the system. Even though this was the first time that CRS had been implemented on a large scale at the university, the surveys showed outstanding results.

5.1 Exciting Results

Faculty members said the clickers increased student engagement, participation, and interaction. They agreed that one of the main benefits of clickers was stimulating class discussion. Students’ opinion of the clickers mirrored what the faculty said. Approximately 70 percent of the students agreed or strongly agreed that clickers made them feel more engaged and increased class participation. Both faculty (80 percent) and
students (66 percent) had an overall positive experience with clickers. Perhaps the most amazing result is the effect of clickers on student grades. The study collected data for 11 courses taught in parallel to assess the impact of clickers on student grades. “T-test of grade data indicates significant difference ($p < .05$) between clicker sections and non-clicker sections. (Kaleta and Joosten 2007, 7)” Even though these are exciting results and reflect those of the qualitative surveys given to faculty and students, these results must be taken with a grain of salt. The use of clickers can unintentionally alter the teaching style which may indirectly increase student performance. For example, since a professor needs time to ask a clicker question, the slow-down in lecturing allows students to pause and rethink what they have just learned and thus has an effect on their understanding without the need for the clicker question. Nonetheless, the results of this study show that students do score better on exams when enrolled in a course that uses clickers.

5.2 Problems that Can Arise

The study also cites the main problems encountered with the implementation of clickers in the classroom. Faculty reported that the learning curve for the software was steep, but that the training provided was very helpful. Perhaps one of the biggest issues with using clickers is the time consumed in class. “Integration of clickers into the design of the class required a greater amount of time than many instructors had anticipated. Also, clicker activities consumed a considerable amount of class time, especially if discussions were linked to questions posed. (Kaleta and Joosten 2007, 8)” Lack of technical support can often be another issue. When a university is piloting a CRS it will most likely not have a team of professionals devoted to working with students and faculty
to solve technical problems. Faculty members at Wisconsin were often forced to deal with students’ clicker issues in-class which cost precious teaching time. Lastly the cost of clickers was cited by students as a concern. However, the students said that significant clicker use in class as well as making clicker responses part of the course grade would help warrant the money spent on them. Here at Dartmouth, students will often not purchase a clicker simply because of the cost since clicker use does not affect participation grades. Because of problems with technical support and the costs, universities are encouraged to choose one CRS and stick with it. This will make providing technical support easier as well as decrease costs because the software and input devices will be purchased in larger numbers.

6 Evaluation of Our System

The system we developed was used in the Spring 2009 offering of CS37 (Computer Architecture). The course was taught by Professor Smith and had 14 enrolled students. CS37 is a mid-to-upper-level course so most of the students in it were either computer science (CS) majors or minors. We hoped that these CS students would be more willing to explore a new software tool and would have a small learning curve. Dartmouth College uses a quarter system consisting of 10-week terms. Professor Smith began using the system approximately half-way through the term. We would have liked to use it for an entire term but we ran into a few complications.

At the end of the term we sent out a survey to the students asking qualitative questions about their experience with the system and for comments and ideas. We chose to ask for open ended feedback as opposed to using Likert scale questions because of the
small sample size. We believed there were not enough students to generate significant statistical results and an open-ended question format would allow students to clearly convey their experiences and ideas. Students agreed that system was useful as an in class tool. Many said it was distracting at first but as the term went on, they became more accustomed to responding to questions. Many of the students did not take advantage of the OUT OF CLASS features both because I neglected to properly demonstrate all the features and there weren’t many questions asked throughout the term. Students said that the OUT OF CLASS part of the system would have been more useful as a review tool if more questions were asked, especially from the earlier parts of the term. The type of questions asked could also have an effect on whether students felt the need to review them. If the questions were simple and geared towards making sure everyone was following along, then students probably wouldn’t feel the need to review them.

However, if the questions were challenging and spurred a lot of class discussion then students would be more likely to go back to them to make sure they still understood the harder concepts. Both the students and the professor desired a better way to generate questions on-the-fly in the classroom that weren’t necessarily recorded in the system.

Students also had a lot of suggestions for improving the Confusion Meter. Almost all the students said that the auto-reset feature was unnecessary. Professor Smith and I had added it thinking that students would forget to change their value back to ‘OKAY’ once they were ‘unconfused.’ However, all the students reported that they would rather have the confusion value not reset. Students also noted that the continuously refreshing confusion meter was distracting. This could be remedied by implementing it differently or only displaying it when a significant number of students
are not at the OKAY value. One student suggested displaying the confusion meter in the student’s UI as well.

Professor Smith did not find the Confusion Meter as useful as he had anticipated. This may be due to the fact that it was underused by the students or that the class wasn’t large enough. He also noted that the complete privacy theme used in the system became an issue with the Confusion Meter and questions. He wanted a feature that would let him select a student randomly from a set with a given Confusion Value or a response choice in a question to explain his or her thinking. This is very similar to the strategy that my physics professor used with his home-grown CRS discussed earlier. This addition would certainly allow the professor to more effectively determine why students are confused or are answering questions incorrectly. However, the lack of privacy could result in students being less willing to set their Confusion Value or respond to questions because of fear of being called on unexpectedly.

7 Future Work

Thus far, we have developed and briefly tested a functioning CRS that incorporates a set of features not present in any commercial CRS. Still, there is plenty more that we would like to see developed and tested.

7.1 Development

We would first like to see the suggestions of the students and professor considered and possibly implemented. Beyond that, would also like to see the system interface with the Blackboard WS mentioned earlier. This would let us take advantage of
all the available course enrollment data. We would also like to see the system interface with WebAuth, Dartmouth’s authentication system. This would eliminate the need for users to remember another set of usernames and passwords and would even more importantly eliminate the need to manually enter login information for all the users for a class. This was not such a problem with CS37 because there were only 14 enrolled students. If the system was to be used in an introductory biology class with over 150 students, entering all of the information by hand would be impractical. We would also like to see a native iPhone client application developed for the system. Both this and the Blackboard WS would be a good demonstration of the usefulness of WS. It would also be interesting to develop more interactive ask-respond designs using either Java or Macromedia Flash. This would be similar to the *Numina II SRS* but would still follow the WS paradigm.

### 7.2 Testing

After fixing the minor bugs in the system and perhaps optimizing the code to perform with a larger number of simultaneous users, we would like to see the system used in a larger classroom. CRS were created for large lecture halls and we believe that our system will have a better impact on the educational experience in classes of around 100 students. It may also be better to use it in a course where professors have used other CRS (such as TurningPoint) previously. These professors likely know the best ways to weave questions into lecture and use the CRS in the most effective ways. We would also like to administer a more detailed survey, several times throughout a term. We hope that
our system increases student performance on exams but this would be challenging to test at a small school like Dartmouth.

8 Conclusion

We began by exploring the commercial classroom response systems (CRS) available and identified improvements we could make. We made a system that is based on web services so a client could easily be written for many input devices. We added a Confusion Meter feature to allow the professor to gauge the appropriateness of the teaching pace without the need to continuously ask questions. Lastly we added a set of features that would allow students to work with questions outside of class as a way to review and further discuss. We were fortunate to be able to test the system in a real class setting thanks to Professor Smith and the students of CS37. The survey sent out to the class and the professor at the end of the term identified the successes and improvements necessary in the system. Using a small class as a test environment allowed the process to be more maintainable though we would like to see the system used in a larger course. We see potential for more development that can be done in the system, both improving the existing features and adding new ones. Overall, the project was an exciting journey into the world of CRS, how they impact the learning environment, and how to harness technology to build the next best system to enhance the teaching and learning experience.
Acknowledgements

Special thanks to my advisor Professor Sean W. Smith for providing guidance throughout the project and using it in his class. I would also like to thank Mark A. O’Neil of the Academic Computing group for sharing his ideas and expertise in writing educational software. Lastly, I would like to thank the students of CS37 for giving my system a test-run and reporting on their experiences and providing ideas and feedback.

Code Availability

Please contact Oleg Seletsky by email to obtain a copy of the code.
Works Cited


Appendix A: Descriptions of Implemented Web Services

The following is the list of web services (WS) that we implemented for the server-side of our system. Each description explains where on the client-side the service is called and the service’s main functionality. These are the services that clients written on other platforms could call to handle all the internal system logic.

**processUserLogin**

This service is called when a user logs in. It checks validity of username/password combination and returns UserId, UserToken, and UserRole if validation was successful. Otherwise it returns a soap fault.

**getListOfCourses**

This service is called when a user's credentials are accepted after login. The list of courses that the user is enrolled is fetched, allowing the user to select one for the session.

**createQuestion**

This service is called when an INSTRUCTOR creates or edits a question. If a QuestionId is provided, then *that* question is 'replaced' with the contents of Question. The intention is that the client will have auto-filled fields when an INSTRUCTOR wants to edit a Question.

**setQuestionStatus**

This service lets the INSTRUCTOR set the STATUS of a question (to OPEN or CLOSED). The service is used when OPENING and CLOSING a Question for polling. If a Question that had been previously used is opened for polling, all previous user responses are deleted from the QUESTION_RESPONSES table.

**answerQuestion**

This service is called when a user, most likely a STUDENT, responds to a Question. The Question must be OPEN for the response to be processed. Otherwise, a fault is returned.

**getListOfQuestions**

This service retrieves a list of Questions from the QUESTIONS table. Through its input parameters, it allows searching the QuestionName and QuestionText fields. It also
allows retrieval of full Questions (ie with QuestionChoices and Files or not) or just the 
QuestionName and QuestionText. It allows a user to specify retrieval of INCLASS 
and/or OUTOFCLASS Questions. If the QuestionList parameter is empty, the service 
returns all Questions for an INSTRUCTOR in the given course_id. For the STUDENT it 
returns all the Questions except for the INCLASS Questions that the STUDENT did not 
answer.

getOpenQuestion

This service gets the full currently OPEN Question. It is very similar to the 
getListOfQuestions service and maybe should be combined.

processSessionVars

This service can be called just to check the SessionVars (i.e. cookies stored on the 
client’s computer). It returns a fault if they do not validate.

getNumResponses

This service returns the number of user responses that there are to a Question. This is 
primarily used in the client to get the number of responses for a live counter. This helps 
the INSTRUCTOR determine when enough students had responded in order to close the 
Question and end polling.

getAddressDistrib

This service returns a list of ChoiceId, NumResponses pairs. It is primarily used in the 
client to create the user response distribution graph.

getUserInfo

This service returns information about a user. Currently this is primarily used in the 
banner of the client. Since the system will most likely be anonymous, this service 
probably won’t be used much.

getCourseInfo

This service returns information about a course such as the department, course name, 
course number, etc.
getPosts

This service returns all the posts for a given Question.

addPost

This service adds a post for a given question by storing it in the POSTS table.

setUserConfusionValue

This service sets the confusion value for a user by storing it in the CONFUSION_VALUES table.

setQuestionArchivedStatus

This service sets the ARCHIVED_STATUS field in the QUESTIONS table to either ‘ARCHIVED’ or ‘UNARCHIVED.’ Only an INSTRUCTOR can call this service. When a question is archived, it cannot be viewed by STUDENTS.

createLecture

This service creates a new Lecture by storing it in the LECTURES table. Each Question must be associated with a Lecture through the lecture_id column in the QUESTIONS table. Only an INSTRUCTOR can create Lectures.

getListOfLectures

This service returns a list of Lectures for a given Course. It returns them in decreasing lecture_start_date order to make recent lectures more accessible (for fetching Questions and adding Questions to Lectures during Question creation).

getConfusionValueDistrib

This service returns a list of ConfusionValue, NumUsers pairs that are used to create the Confusion Meter graph.