WUTexter: A Classroom Interaction Tool For Anybody Who Can Text

Benjamin Murray, Hunter LaTourette[1], and Ron K. Cytron[1]
[1]Department of Computer Science and Engineering, Washington University, St. Louis, Missouri, USA

Abstract— *The classroom experience is enhanced for both teacher and student alike when feedback mechanisms are in place to measure the students' engagement. Specialized devices such as the* `iClicker` *have been developed along these lines, but these devices have limited range and use. This paper describes the design and implementation of a device that can be used within reach of the Internet or cell tower, allowing students to interact with the instructor in the classroom by sending simple text messages. The application currently accommodates polls, student-supplied questions, and a mechanism to express confusion or boredom. We provide some early anecdotal experiences using this device, articulate plans for the future, and describe how others can use this technology in their own settings.*

Keywords: classroom interaction, texting

1. Introduction

The context for the work described in this paper began with our department's decision to purchase 50 `iClicker` [1] devices for use in our department's courses. While many schools require or encourage (through extra credit) each student to purchase his or her own `iClicker`, we opted for a departmental basket containing 50 of these devices. An example of such a device currently in production and for sale is shown in Figure 1. An instructor would bring these devices to class, distribute them randomly among the students present, and construe the results of their use as statistically valid random sampling of the (perhaps larger) classroom population.

These plans did not account for the combined battery weight of 50 such devices. With two batteries per device, the basket of `iClicker`s was too heavy to carry easily across campus. Upon hearing this, we began to consider how students could participate in feedback such as that sought by the `iClicker`, but by using technology already in their possession. It occurred to us that all students we knew could *text* using devices that connect to the cell networks. Most students are exceptionally adept at texting, owing to the frequency they communicate with friends and family using texting. Many, if not most, of their devices can also connect to the Internet.

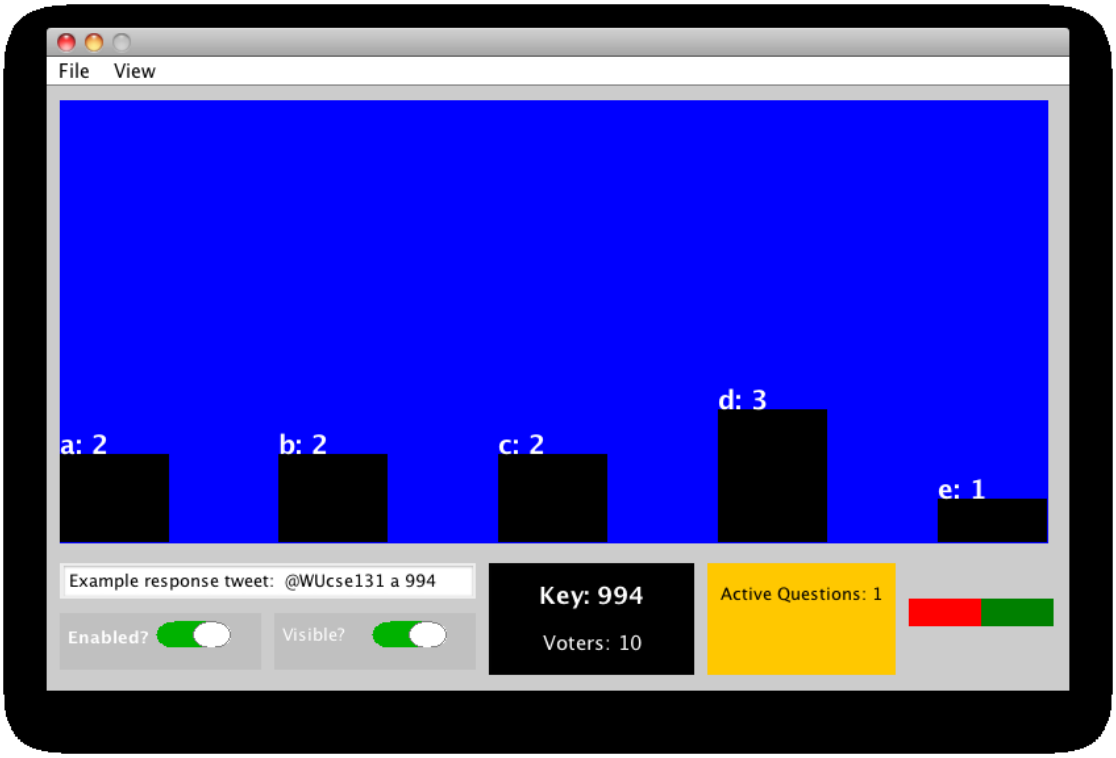
While our initial motivation for the work we present in this paper was to free our instructors from carrying the

Figure 1: An `iClicker` device. Five buttons are provided for its user to respond to polls. From `http://www.iclicker.com`.

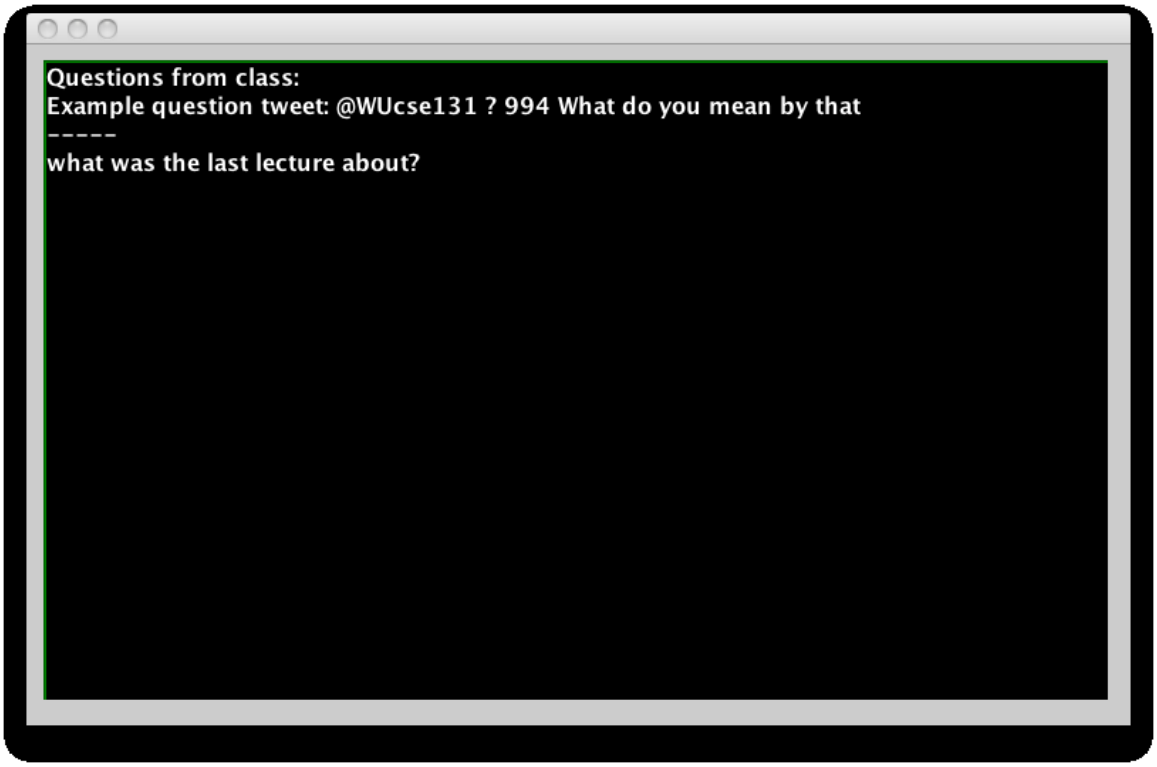
`iClicker`s to class, we arrived at the following arguments in favor of our approach:

- `iClicker`s require a base receiver for operation. This base receiver is purchased separately from the `iClicker`s, and it must be brought to class and connected to a computer to receive and tabulate students' responses.
- `iClicker`s are limited in transmission range. Due to FCC regulations, the `iClicker` devices are limited in the strength of signal they are allowed to emit. The devices and base station work within a room, but they cannot work for students situated remotely from the base receiver.
- `iClicker` devices are very limited in the form of response. Referring to Figure 1, a student presses one of five buttoms to register response. In particular, these devices lack the ability to *initiate* interaction with the instructor, to pose a question or to indicate confusion or boredom.

While the design and implementation are described in greater detail below, we begin with a quick description of the features of `WUTexter`. The application launches into the two panels shown in Figure 2. The main panel (a) controls the application and displays almost all of the information. The other panel (b) serves to display only the text of questions posed by the participants. This separation into two panels is convenient in terms of how the information



(a)



(b)

Figure 2: (a) The primary `WUTexter` control panel and display. (b) Participants' questions are displayed in a separate panel.

is displayed in the classroom. An instructor can choose to place the students' questions panel on a private display, while still showing the results of polls on a screen seen by the participants. The components of the display shown in Figure 2(a) are as follows:

- The largest portion of the window is dedicated to showing the results of polling. Each vertical bar indicates by its height the fraction of participants who have chosen that particular response. The example shown here represents 10 participants' responses to a multiple-choice poll offering 5 possible responses. As shown, responses `a`, `b`, and `c` have each received two votes apiece. Response `d` earned 3 votes and response `e` earned only 1 vote.

  This portion of our tool replicates the functionality found in the `iClicker`. A small difference is that the number of available choices can be configured when the tool is launched, with the default of 5 to match the `iClicker`.
- Continuing down the screen we see an example tweet to help students model their responses:

  ```
  Example response tweet:  @WUcse131 a 994
  ```

  The response tweet contains three pieces of information:
  - The `twitter` account name for the class, appended to an at-sign (`@`).
  - The student's desired response (`a`, `b`, etc.).
  - A 3-digit secret key for this `WUTexter` session (Section 3.5). This key makes it difficult for those not privy to the lecture to participate in the `WUTexter` session. The key can be changed at will by the instructor.
- Shown below are two sliders that control whether the poll is enabled (open) or not (closed), and whether the polling results are dynamically visible to the audience.
- Continuing to the right, a black box at the bottom shows the current secret key (994 in our example) and tabulates the number of participants in the current poll.
- Continuing to the right, a box indicates by its background color whether questions have been asked by the students. The background remains a bright color until the instructor clicks on the box to reset the tally to 0. At that point, presumably the instructor has addressed the posted questions or has chosen to ignore them.

  A question is posed when a participant substitutes a `?` for a poll choice such as `a`, in the position where the poll choice would normally appear. As before, the secret key must be the next token supplied on the response. The remaining text is interpreted as the participant's question and is posted on the panel shown in Figure 2(b).
- The final component to the right is a red and green bar that reflects the participants' view that the material should be delivered more quickly or slowly. A student does this by furnishing a `+` sign instead of a letter in the model response. Similarly, a `-` sign requests that lecture slow down. This is described in greater detail in Section 4.3.
- The `File` and `View` menus allow instructors to access other functionality, such as resetting the key, zeroing the tabulations, and exporting a log file of those who have participated in the session.

While in-depth studies of `WUTexter` have not been conducted, we report on the tool in this paper so as to make it available for more widespread use.

2. Related work

The use of mobile devices in the hands of students and instructors is currently a topic of great conversation in academia [2]. Not surprisingly, computer science courses are among the first to adopt and adapt to such technology [3]. Tim Hickey and others [4] have considered student feedback to affect the pace of the classroom. They have also developed a system that allows peer-instruction during class, so that students can field and answer other students' questions.

3. Approach

Seeking a method by which any student, anywhere, could participate in an `iClicker`-like poll, it became clear that we would need to rely on the Internet or the cell phone network for communication. Broadly speaking, we required an *event channel* [5] that could relay messages from students to an application that could tabulate results, report students' questions, and keep track of the students' confusion or boredom.

Many social media sites allow messaging among their participants. However, privacy concerns typically limit such messaging to those who have explicitly allowed such activities. Moreover, we do not need these messages to persist, so it would help if the messages are easily deleted by the receiver (our application). `Facebook` surfaced early as a candidate, but students understandably protect their `facebook` presence. Also, `facebook` distinguishes messages sent between friends and non-friends. We could not require students to be friends with the class account. We soon turned to `twitter` for the following reasons:

- `Twitter` is primarily concerned with messaging, and other features are less prominent. For `facebook`, messaging is not its primary mission.
- It is easy to create `twitter` accounts that use the same `gmail` account. Thus, a professor could have separate accounts for each class without having to create a new email address for each one.
- The API seemed easy to use. An application could generate authentication keys, which can then be used by an application to create a `twitter` session.

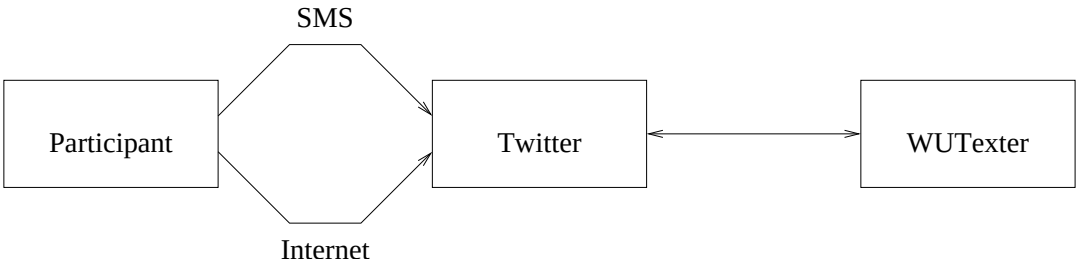


Figure 3: Software architecture of `WUTexter`.

- `Twitter` can use `SMS`, through which text messages of very simple syntax can simulate tweets. This puts our approach within reach of any student who has access to the Internet or who can text, even from a very simple phone.
- While `twitter` does limit any user to sending no more than 250 messages per hour, there is no such limit on the messages an account can receive.

Figure 3 shows the software architecture of `WUTexter` using `twitter` as an event channel. The participant reaches `twitter` using either the cell tower via SMS (Short Message Service) or over the Internet. To use SMS, the participant simply sends a text message to a special SMS "phone number" (40404 for `twitter`). The body of the text message is of the format described previously. If the participant has access to the Internet, perhaps using a data plan on a mobile device, then he or she can reach `twitter` in the usual way, by logging into `twitter` via a browser or by accessing his or her `twitter` account using an application.

Once the participant has tweeted a message, the `WUTexter` box in Figure 3 picks up such messages by listening for references to its account name. In Figure 2, the account shown is `WUcse131`. The case-spelling of the account name does not matter, but the capitalization serves to show how the account name was chosen: WU for Washington University and the rest of the account name is a course at the university.

Any `twitter` account can be used, but it is helpful to choose an account for a course whose name is easily remembered. The tweet message must mention the account name (recall `@WUcse131`), and the `WUTexter` application must access `twitter` by authenticating to the same account. A personal `twitter` account could be used, but we find it convenient to dedicate an account to a particular course. That same account is then used each semeter regardless of who teaches that course.

3.1 Instructor preparation

To use `WUTexter`, an instructor must decide on a `twitter` account to be used as the target of messages sent by the students. As described above, we establish a separate `twitter` account for each course using `WUTexter`. For example, our CSE131 course has the `twitter` account `wucse131`. Accounts named in this way should be readily available at `twitter`. Generally, the following steps are needed to set up such an account:

1) On the `twitter` home page, request an account with a suitable name.
2) Each `twitter` account needs an email address to which it is associated. If the instructor already has a `gmail` account (say, `professor@gmail.com`), then an email address of the form

   `professor+wucse131@gmail.com`

   can be used, because `gmail` disregards the portion at and after the `+` sign. Thus, confirmation (and any other desired) messages will be sent to the instructor's specified `gmail` account.
3) An instructor typically has no reason to see the email messages generated by `twitter`. The preferences section of a `twitter` account can be visited and email notifications can be disabled. In practice, this results in almost no messages at all being sent to the associated account.
4) Using the created and verified `twitter` account, log into `dev.twitter.com`. Here, you will generate an application tied to the account.
5) Finally, generate authorization keys so that the `WUTexter` application can log in under your new account to receive tweets from students. The text of these keys is saved in a configuration file used by `WUTexter`

3.2 Student preparation

For the student side of this application, a `twitter` account is needed, and the tweets of that account must be publicly visible (such is the default for a new `twitter` account). Students may decide to use their personal or otherwise extant `twitter` accounts, or they may decide to establish a new one specifically for use with `WUTexter` or for a given course. In terms of participating in `WUTexter` activity, there is no difference.

However, some instructors may elect to give credit based on participation, in which case the association of

a `twitter` account with a student must become known so that credit could be given. `WUTexter` can save log files from its sessions, and the correspondence between a `twitter` account name and a particular student must be discovered at some point to award credit. In our courses, we typically keep this correspondence secret until the end of the semester, so that students can use `WUTexter` anonymously. We find that students are more likely to pose questions if their identity is secret. At the end of the semester, students can tweet their student ID at the class `twitter` account, so that credit can be awarded for participation if that is the instructor's policy.

3.3 `Twitter` as an event channel

Messages related to `WUTexter` are sent from student to the classroom tool via `twitter`. Recalling the form of a message, the following

```
@wucse131 c 377
```

sends a message to vote for option `c` using secret key `337`. The message is intended for the `wucse131` `twitter` account, and the at-sign is necessary in front of the account name so that `twitter` alerts the class account on behalf of this message.

Although a large number of students may be voting at once, each is using his or her own `twitter` account. Currently each such account holder is limited to some 250 messages per hour, which is ample to accommodate the intended usage here.

The receiver of these messages, the `wucse131` account in the above example, cannot be held responsible for those *mentioning* the `wucse131` account name in a tweet. There is thus no limit, from what we have seen, in terms of the number of messages the account can receive.

There are two ways in which messages can be sent:

- **tweeting** The examples have thus far used this mechanism. The account is mentioned using the at-sign notation, which causes the message to be sent to the account as a *status* update.
- **direct message** `Twitter` allows its users to send direct messages to each other. A student can send a direct message to the class account, providing that the class account is *following* the student's account. Thus, once per session, our in-classroom `WUTexter` tool responds to a status message by *following* the source of that message, thereby making sure that direct messages are possible.

Why use direct messages instead of tweets? There are two reasons:

- A direct message is private, and would not be displayed in a user's `twitter` feed.
- A given tweet can be sent only once. Consider a student participating in a poll, who initially feels `a` is the correct response. If that student changes his or her mind to `b` and then back to `a`, the last tweet to assert the response as `a` is redundant and is therefore suppressed by `twitter`. The last nonredundant message would the final say for such a student, so it would appear this student voted for `b`.
  A direct message can be sent twice, so that mechanism allows students to change their minds more frequently and fluidly.

In terms of ease of use, tweets are slightly shorter than direct messages. The improvements we have in mind for the student-client would automatically favor direct messages over tweets. This is discussed in Section 5.

3.4 Running the in-classroom server

There are two items necessary to run the server:

- A copy of the `WUTexter` jar file.
- A configuration file

The jar file contains the code as described below. The configuration file contains the authorization keys necessary for a `twitter` account to open a session with `twitter`. The server by default opens the file `config.txt` as its configuration file, but this can be changed when the server is started. It is probably most convenient simply to store a given class's configuration file as `config.txt`.

The `WUTexter` server that runs in the classroom is written entirely in Java, and uses the `twitter-4j` library to access the `twitter` API. The resulting code is packaged in a jar file so that it can be run in any classroom that supports Java. Double-clicking the jar file icon typically suffices to launch it. From a command line prompt, the following also runs the server:

```
java -jar WUTexter.jar
```

With the application written in Java, we have seen no problems at all running the server in classrooms across our campus. The server can be run on an in-classroom computer (Java is typically present) or on the instructor's laptop.

3.5 Secret key

Each `WUTexter` session creates a 3-digit key. Messages that are missing the key are ignored. The purpose of this key is to ensure that only those students who are participating in the class (locally or remotely) can send messages that are considered by the application. While the logical basis for the key is sound, the key requires extra work on the part of the student, and we return to this point in Section 5.

4. `WUTexter` modes and experience

The `WUTexter` tool has been used in several classes at Washington University, with enrollments from 30 to over 300. In rooms with multiple projectors, it is possible to have the display shown continuously on one screen while lecture materials are presented on the other screen. For

rooms with a single projector, many laptops offer the ability to show different images on the laptop and the projected screen. In such cases, `WUTexter` could be running in the instructor's view, and dragged to the students' view as needed. As seen in Figure 2, there are various and concurrent ways in which `WUTexter` can be used, as described below.

4.1 Polling

In terms of its conceived design as a replacement for `iClicker`, the most apparent mode of operation for `WUTexter` is to offer students a mechanism for responding to a poll. The purposes for such polls include the following:

- How prepared are students for class? Did they read the assigned materials?
- How well are students following the lecture material thus far? Does some material need to be covered again?
- How well can students apply what they have learned in new settings?

Thanks to the `iClicker`'s widespread availability and use, others have considered ways in which they can be used for conducting polls [6]. The mechanism supplied here is identical to `iClicker`, except that students can participate remotely and use only their cell or smart phones to do so. The experience so far with polling has been positive, but we remark in Section 5 on improvements we can make client-side that would make this and other functionality easier for students to use.

4.2 Posing questions

Students typically have questions, and many of these students are reluctant to ask questions in class for a variety of reasons:

- They don't want to interrupt the lecture.
- They don't want to disclose their uncertainty, confusion, or misunderstanding.
- They may believe they are the only ones who misunderstand, so they are reluctant to take up class time to get an answer to their questions.

As a result of the above disincentives, many students *disconnect* with lecture at the point where they have a question but feel unable to pose the question to the instructor. This disconnection may occur because the given student's mind is preoccupied with the question, or it may be the case that because of the question, the student feels (perhaps hopelessly) lost.

While designing the `WUTexter` tool, we realized that we could accommodate information well beyond that of a poll and its responses. A channel was effectively open by which students could ask questions anonymously. The questions would appear on the screen, so that a professor need not stop mid-sentence. The professor can decide whether and in what manner to answer the question.

When deployed in a classroom, this feature seemed to earn the most attention and praise. `WUTexter` can be enabled as students are entering the class, some 10 minutes before class officially started, and students fell into the habit of sending in questions right away. This allows instructors to respond to common concerns before class started. Students reported that they really liked and appreciated the anonymity with which they could pose questions. With students having an essentially open forum to express themselves, it would have been possible for them to send annoying or inappropriate comments, but this never occurred.

4.3 Confuse-o-meter

There are occasions where a student would like lecture to speed up or slow down, without having a particular concern or question to pose. For such students, we offer the confuse-o-meter, which could have the following interpretations:

- Should the lecturer speed up, keep the current tempo, or slow down?
- Is the class generally fine, confused, or bored?

Students can send messages via `WUTexter` to cause the confuse-o-meter to deflect according to their mood. The display in Figure 2 is a horizontal green and red bar. With equal amounts of each color, the class is fine. As the bar becomes more green, the lecturer should increase the pace, and as the bar becomes more red, the lecturer should slow down. The bar has a return-to-center behavior over time, so that if students do not keep sending messages, it decays back to a normal, middle setting.

While students initially found this feature intriguing, and they played with it a bit at the beginning of the semester, it did not prove that useful during the course.

5. Conclusions and Future Work

While we have found `WUTexter` to work well in our courses, we discuss in this section some simple changes that would make the application easier to use.

- The form of the messages appears simple, but on most phones the messages require changes between character sets (letters, numerals, special symbols). The course name contains numerals, the account must be prefaced with an at-sign, and the main text of the message is typically alphabetic.
- The secret key, while useful in preventing intrusion, is a burden for students to remember and type.

We are currently addressing the above problems by developing a client-side app for the iPhone and the Android platforms, along the following lines:

- The app would sense whether messages should be sent via data plan (Internet) or texting (cell tower) based on which resources are available and which method is preferred by the user.
- The app could present the 5 buttons of the `iClicker` as shown in Figure 1, a text area for asking questions, and a $+$ and $-$ button to indicate confusion or boredom.

- The app could make some commonly asked questions available from a menu, so that typing them would be unnecessary.

Acknowledgments

The authors thank Roger Chamberlain for his feedback concerning the use of this tool. The authors also thank the many students who have shared their reactions and experience using `WUTexter`. We thank the reviewers for their feedback.

References

[1] "iclicker web site," http://www.iclicker.com.

[2] A. Czapracki and C. Burrows, "Bringing students and faculty together through mobile devices," in *Proceedings of the 39th Annual ACM SIGUCCS Conference*, ser. SIGUCCS '11. New York, NY, USA: ACM, 2011, pp. 215–218. [Online]. Available: http://doi.acm.org/10.1145/2070364.2070420

[3] A. T. Chamillard, "Using a student response system in cs1 and cs2," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 299–304. [Online]. Available: http://doi.acm.org/10.1145/1953163.1953253

[4] T. J. Hickey and W. T. Tarimo, "The affective tutor," in *Proceedings of the 2014 meeting of the Consortium for Computing Sciences in Colleges, Northeastern Region*, 2014.

[5] T. H. Harrison, D. L. Levine, and D. C. Schmidt, "The design and performance of a real-time corba event service," in *Proceedings of the 12th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, ser. OOPSLA '97. New York, NY, USA: ACM, 1997, pp. 184–200. [Online]. Available: http://doi.acm.org/10.1145/263698.263734

[6] T. Murphy, K. Fletcher, and A. Haston, "Supporting clickers on campus and the faculty who use them," in *Proceedings of the 38th Annual ACM SIGUCCS Fall Conference*, ser. SIGUCCS '10. New York, NY, USA: ACM, 2010, pp. 79–84. [Online]. Available: http://doi.acm.org/10.1145/1878335.1878356