Administering Breaks to Students during Tutoring Sessions: A Human-Robot Interaction Study

Arsalan Sufi
Aditi Ramachandran and Brian Scassellati
5 May 2016
1. Abstract

The benefits of one-on-one tutoring are well-documented. Unfortunately, many schools lack the resources to provide one-on-one tutoring to all students. Intelligent tutoring systems, or ITSs, could provide a viable alternative to one-on-one human tutors. The behaviors of human tutors are nuanced. In addition to administering content, gauging mastery, answering questions, and engaging in other “task” behaviors, human tutors greet their students, encourage them, and converse with them. Determining how ITSs can improve learning outcomes by also performing “non-task” behaviors is an interesting undertaking. My project specifically focuses on administration of breaks to students. The attention spans of students listening to teachers can be as short as 5 minutes, and breaks can provide students with much needed cognitive rest. Break provision is nuanced however. For example, breaks need to be timed properly if they are to be as effective as possible. My project involves three components: a literature review, a series of experimental designs, and two fully implemented break activities embedded within an ITS for an upcoming study in the Yale Social Robotics Lab. The literature review covers ITSs, non-task behaviors, and breaks. In terms of experimental designs, two full-fledged designs are explored. The first focuses on the frequency of breaks and builds off of the hypothesis that breaks are beneficial in moderation, detrimental in excess. The second focuses on personalization of break provision, in other words, having the robot determine when breaks should be administered using student timing and accuracy data. Finally, the two break activities that I’ve implemented include a tic-tac-toe game and a stretch break with the Nao robot.

2. Background and Preliminary Research

2.1. Intelligent Tutoring Systems

The benefits of one-on-one tutoring are well-documented. Students who receive one-on-one tutoring perform better than students who learn via conventional classroom instruction [1, 2]. Unfortunately, many schools lack the resources to provide one-on-one tutoring to all students. Intelligent tutoring systems, or ITSs, could provide a viable alternative to one-on-one human tutors. ITSs are computer-based learning systems that adapt to the needs of learners. Unsurprisingly, many ITS studies focus on personalization. Human-robot interaction enters the picture when ITSs use robots as learning agents.

2.2. Non-Task Behaviors

The behaviors of human tutors are nuanced. In addition to administering content, gauging mastery, answering questions, and engaging in other “task” behaviors, human tutors greet their students, encourage them, and converse with them. Determining how ITSs can improve learning outcomes by also performing “non-task” behaviors is an interesting undertaking. The non-task behavior that I’ve focused on this semester is administering breaks to students. Before studying breaks however, I want to take a closer look at non-task behaviors in general.

2.2.1. Non-Task Behaviors and Positive Effects
ITSs that engage in socially supportive behaviors have been shown to improve learning outcomes and student experiences. Saerbeck, Schut, Bartneck, and Janse conducted an ITS study with children ages 10 to 11 and the iCat robot [3]. The children were divided into two groups. One group worked with a neutral robot and the other with a socially supportive robot. The neutral robot administered content and did little else whereas the socially supportive robot performed various non-task behaviors. It nodded, guided students using its gaze, and expressed emotions. The students who worked with the socially supportive robot performed better on the study’s post test than those who worked with the neutral robot. They also exhibited higher intrinsic motivation as determined by a questionnaire.

2.2.2. Non-Task Behaviors and Negative Effects

The relationship between non-task behaviors and student experiences isn’t always positive however. Veletsianos conducted an ITS study with adults in a master’s program and a digital learning agent [4]. The participants were separated into three groups: off, on/off, and on. Like the neutral condition in the Saerbeck study, the off condition was void of non-task behavior. In the on/off condition, the digital character made some non-task commentary, and in the on condition, the digital character made a considerable amount of non-task commentary. Veletsianos found that there was no statistically significant difference in learning outcomes between the off and on/off conditions, and that there was in fact a decrease in learning outcomes in the on condition.

2.2.3. Explaining These Different Effects

Various factors explain the different outcomes of the Saerbeck and Veletsianos studies. First, Saerbeck worked with a robot and children whereas Veletsianos worked with a digital character and adults. (The intersection of robots and children will be explored in greater detail in Section 3.1.1.) The other critical difference between the two studies is the nature of the non-task behavior. The adults in the Veletsianos study noted that the digital character’s behaviors seemed “rehearsed” and “fake” (p. 280). This is partially explained by the fact that the digital character made its non-task commentary at predefined points in time. This contrasts with the Saerbeck robot, which acted in response to events and utilized a state-based model of behavior. For example, the children could activate the robot by tapping it.

This leads to an important point: non-task behaviors are more effective when they are administered in response to a student’s needs, that is, when they are personalized. Many HRI papers emphasize the importance of social-bond formation during successful human-robot interactions [3, 5]. Personalized non-task behaviors can facilitate the formation of these bonds, as was the case in the Saerbeck study, where the students perceived the robot “more as a friend than a teacher” (p. 1619). Less personalized non-task behaviors, on the other hand, can actually inhibit the formation of these bonds, as was the case in the Veletsianos study, where the students seemed to “actively resist and reject the agents’ human-like features” (p. 280).

A quick note before moving forward: throughout this section, I’ve used the term “non-task behavior” and “socially-supportive behavior” somewhat interchangeably. Although not all non-task behaviors are explicitly social, many seem to have at least some social component.
2.3. Breaks

One important example of a non-task behavior, and the behavior that I’ve focused on this semester, is having students take breaks. The attention spans of students listening to teachers can be as short as 5 minutes [6]. For older students (undergraduates for example), these numbers are only slightly higher—usually 15 to 20 minutes [6]. The capacity for sustained attention develops significantly between age 11 and adulthood [7]. Accommodating the short attention spans of students, specifically younger students, during learning interactions is thus incredibly important. One way of accomplishing this is to have students take breaks.

2.3.1. Framing Breaks in Terms of Cognitive Rest

Research supports the intuitive idea that breaks are beneficial and provide needed cognitive rest. Brief and rare disengagements from a visual vigilance task can help maintain heightened levels of vigilance [8]. By deactivating and reactivating vigilance goals, vigilance decrements can be avoided [8]. This idea extends to breaks of longer lengths. Breaks of 1, 5, and 10 minutes have all been shown to have restorative effects on performance on an auditory oddball task [9].

All that said, breaks need to be timed properly if they are to be as effective as possible. [9] notes that “task interruptions are beneficial only when they are of sufficient quality and duration and possibly only when given at the appropriate time.” The duration of a break is also important. The same study that found that breaks of varying lengths all have restorative effects also found that focus declines more rapidly after longer breaks [9].

2.3.2. Framing Breaks in Terms of Social Bonds

In addition to framing breaks in terms of cognitive rest, we can frame them in terms of the social bonds that I discussed in Section 2.2.3. If a robot successfully determines that a student is tired and accordingly lets the student take a break, the student will likely feel better understood by the robot. Conversely, if a robot tells a student that she seems tired or frustrated when she really isn’t, the student will likely have a harder time thinking of the robot as a social partner. This social dimension heightens the importance timing breaks properly.

3. Experimental Design and Additional Research

Although our research question has evolved over the course of the past semester, every iteration has centered around the same fundamental question: how should ITSs administer breaks to students during tutoring sessions? Throughout the rest of this section, I’ll describe the evolution of our research question and our corresponding experimental designs, exploring additional scientific literature in the process.

3.1. Shared Features of All of Our Experimental Designs

Despite the differences across the experimental designs that we’ve considered, the designs all have the same underlying structure. The ITS uses a robot as a learning agent, and the participants are children. The participants take a pre-test before the study. They then complete a tutoring session with
the ITS. After the tutoring session, they take a post-test and are interviewed. The conditions always involve the ITS’s behavior with respect to break provision. In Condition 1, for example, the ITS may give students breaks along a fixed schedule whereas, in Condition 2, the ITS may give students breaks when it determines that they’re tired or frustrated. (Again, these are just examples. The actual conditions changed considerably over the course of the semester and will be discussed in subsequent sections.) Every student is assigned to a single condition. In other words, students don’t spend part of their time working with a Condition-1 ITS and the rest of their time working with a Condition-2 ITS; they spend all of their time with a single-condition ITS. This prevents biases from forming over the course of the study. If a student spends the first half of the study working with a highly personalized robot and then spends the second half of the study working with a more static robot, the student may find the second half of the study to be particularly undesirable given the contrast with the first half of the study.

Regarding the ITS setup, it consists of a Nao robot, a tablet, and a few additional components that the student doesn’t interact with. The student is presented with questions and enters answers to those questions using the tablet. Nao supplements the tablet by guiding the student through the tutoring session, narrating questions, and interacting with the student. The architecture of the ITS is described in greater detail in Section 4.1.

In Sections 3.1.1., 3.1.2, and 3.1.3, I’ll describe the rationales for three features shared across our experimental designs: 1) the selection of participants and learning agents, 2) the selection of break activities, and 3) the selection of evaluation metrics. In the reminder of Section 3, I’ll explore the features that varied across our experimental designs by describing the experimental designs themselves.

3.1.1 Participants and Learning Agents

Selection of participants and selection of learning agents shouldn’t happen in isolation. Different combinations of participants and learning agents pose unique opportunities and challenges.

Participants. As noted in Section 2.3, the attentions spans of younger students tend to be shorter than those of older students [6]. As a result, even though breaks can be beneficial for students of all ages, the benefits are magnified for younger students. We’ve accordingly decided to work with children. The specific age of the children will be determined by the problem domain that we select for the study.

Learning agents. Some ITSs use digital, on-screen characters as learning agents while others use physical robots. HRI research has found that individuals experience increased enjoyment and are more compliant when interacting with physical robots than when interacting with on-screen characters [10]. Research also suggests that physical embodiments make it easier for individuals to perceive agents as social interaction partners [11, 12]. We’ve accordingly decided to use robots in our study, specifically Nao robots. Given that I’m completing my senior project in the Yale Social Robotics Lab, this probably doesn’t come as a surprise.

Children and robots. Although there are isolated reasons for selecting children and robots as participants and learning agents, respectively, there are additional reasons that involve the
intersection of children and robots. Many children, especially children younger than age 12, ascribe human characteristics to robots [13]. This contrasts greatly with the opposite intersection of digital learning agents and adults. Although adults humanize these digital characters, they also recognize them as programmed tools and, under certain circumstances, resist their lifelike behaviors [4]. This makes social-bond formation easier with the former, harder with the latter.

3.1.2 Break Activities

In selecting our activities, we’ve had to balance academic and technical concerns. On the academic side, it’s clear that the selected activities should be interactive as this facilitates the formation of social bonds. Given the importance of a robot’s physical presence in a human-robot interaction [11, 12], the activities should also have the robot move and interact with its environment whenever possible. Physicality is ultimately what separates robots from digital characters, no matter how interactive. On the technical side, we face various constraints. For example, even though a conversation break would be incredibly interactive, rapid speech processing is challenging. This limits the ability of the robot to engage in a natural conversation with the student. Inputs to the robot will accordingly have to be restricted to tablet buttons and text fields.

It’s also important to note that we want multiple types of break activities. If just one activity is selected and repeated multiple times, the breaks will likely become less novel and less effective over time. Although we’ve decided to administer different types of breaks, we plan to provide the breaks to every student in the same order so as to minimize differences across conditions.

We’ve fully settled on two break activities; additional activities have yet to be selected.

**Tic-tac-toe break.** The first break involves playing tic-tac-toe with Nao. This activity is highly interactive as the robot is constantly responding to the student’s actions. This interactivity can be amplified with speech; the robot can verbally acknowledge the student’s actions as well as its own actions, using statements like “Nice move!” and “Looks like it’s my turn now.” Regarding technical constraints, the game will be played on the tablet. Nao won’t be holding a pen and writing on a piece of paper.

**Stretch break.** For the second break, we sought inspiration from teachers. Teachers often have their students take breaks to stretch. We’ve accordingly decided to have Nao guide the student through a stretch routine. Like the tic-tac-toe game, this activity is interactive as the robot and student will stretch together, the robot leading the student. This activity also emphasizes the physicality of the robot. The implementations of these two activities will be explored in Section 4.

3.1.3 Evaluation Metrics

The last ITS study run at Yale used a pre-test and a post-test to measure student learning gains. We intend to do the same for this study. It’s important to note, however, that learning outcomes are an indirect measure of our true dependent variable: student engagement. This indirect means of measuring student engagement builds off of the reasonable assumption that students learn more when they are engaged and enjoying themselves.
This indirect approach has its limitations however. If the differences in learning outcomes are minimal across conditions, we shouldn’t immediately conclude that well-designed break provision has no benefits. For example, consider a student who doesn’t perform particularly well on the post-test but was consistently focused during the tutoring session and is keen on studying more with the ITS in the future—this is arguably a success. With this in mind, we’ve tried to think of ways to measure engagement more directly.

The first and most straightforward approach is to have students fill out questionnaires and to interview them after the tutoring session. This can be challenging with children, who may feel pressured to say that they enjoyed working with Nao even if they didn’t. We plan to conduct interviews despite this.

In terms of quantitative measures of engagement, we can track gaze to determine the percentage of time that students spend looking at the tablet or the robot. This approach also has limitations. For example, many students look off to the side when working. Likewise, gaze detection systems can be hard to configure and may reduce the autonomy of the ITS. (If the gaze detection system fails, someone may need to intervene to correct the issue.) Other quantitative measures that are easier to collect include timing and accuracy data. If a student consistently proceeds through questions more quickly and with greater accuracy after breaks, this could indicate that breaks are being provisioned well.

3.2. Experimental Design 1: Focusing on Frequency

3.2.1. Motivation

Our initial experimental design focused on the frequency of breaks. Our hypothesis was that breaks in moderation can increase engagement and improve learning outcomes, but an excess of breaks can distract students and actually worsen learning outcomes. This builds off of the research presented in Section 2. Although a great deal of evidence (and intuition) suggests that breaks are beneficial, the Veletsianos study provides an important counterexample. Students in the on/off condition (moderate levels of non-task commentary) performed at the same level as students in the off condition (no non-task commentary), but students in the on condition (high levels of non-task commentary) performed worse than both other groups, partly because they found the commentary distracting. Given that the only difference between the on/off and on conditions was the higher frequency of non-task commentary, frequency seems to matter. Of course, the effects of frequency may change when the non-task behavior or break is well-designed; one of the conclusions of the Veletsianos paper was that the non-task commentary wasn’t well-designed. Even so, additional research has documented the pitfalls of overstimulation in educational contexts [14].

3.2.2. Description

This first experimental design involved three conditions:

- Condition 1: No breaks (control)
- Condition 2: Moderate break frequency
- Condition 3: High break frequency
The breaks in Conditions 2 and 3 would occur along a fixed schedule, for example, after every 5 minutes or after every 5 problems. We planned to use data on average attention spans to select the specific break frequencies for Conditions 2 and 3. As noted in Section 2.3, the attention spans of students listening to teachers can be as short as 5 minutes [6]. As a result, a break every 5 minutes would be a reasonable frequency for Condition 2. A higher frequency would be selected for Condition 3. To allow for valid comparisons across the conditions, every condition would cover the same number of problems in total. Regarding the problem domain, we planned to work with fractions, as was done in Yale’s last ITS study.

3.2.3. Flaws

There were numerous flaws with this design.

Lack of interesting conclusions. There were two conclusions that we hoped to reach. The first was general; we hoped to validate our hypothesis that breaks were helpful only in moderation. The second was more specific. Related to the aforementioned hypothesis is the idea that an optimal level of breaks exists, a level that maximizes engagement. Given the coarse granularity of the design, it wasn’t intended to find this optimal level; rather, it was intended to begin the search for this optimal level. For example, if Condition 2 resulted in the greatest learning gains, a follow-up, finer-grained study could be conducted with break frequencies closer to Condition 2’s.

Both of these potential conclusions have limitations. The issue with the first potential conclusion is that it isn’t particularly interesting. Most educators would agree that breaks can be beneficial but distracting if used too often. The issue with the second potential conclusion is that it’s hard to separate the conclusion from the context in which it was reached. That is, we wouldn’t be finding the optimal break frequency for ITSs in general; we’d be finding the optimal break frequency for fraction problems and our selected break activities.

Lack of a clear independent variable. The independent variable in this design, break frequency, actually varied multiple aspects of the tutoring session at once. Assuming that the total amount of content covered remains the same and that every break has the same duration, increasing the frequency of breaks simultaneously increases the total amount of time spent taking breaks as well as the total duration of the tutoring session.

Figure 1 (on the next page) makes this clear. The blue intervals represent on-task time, and the red intervals represent breaks. By increasing the frequency of breaks, the total amount of time spent taking breaks increases from 2 minutes in Condition 1 to 4 minutes in Condition 2. Likewise, the total duration of the tutoring session increases from 14 minutes in Condition 1 to 16 minutes in Condition 2.
As a result of this, it may have been hard to determine what exactly was responsible for differences in learning outcomes, if any were observed. Our independent variable wasn’t perfectly isolated.

3.3. Experimental Design 2: Focusing on Personalization

3.3.1. Addressing Design 1’s Flaws

To address the flaw in Design 1’s independent variable, we selected a new independent variable for Design 2: timing of breaks. In other words, we elected to keep the number of breaks fixed across the new conditions and, instead, decided to vary the method of delivery. The new conditions were the following:

- **Condition 1; personalized**: The robot determines when to administer breaks. Engagement will ideally be the highest in this condition.
- **Condition 2; fixed**: The robot administers breaks along a fixed schedule.
- **Condition 3; random**: Breaks are randomly dispersed throughout the tutoring session.

In order to keep the number of breaks fixed across the conditions, Condition 1 needed to be modified slightly. It specifically needed to be split into two phases. Let’s say we wanted to deliver \( n \) breaks. During the first phase, the robot would decide when to administer breaks. During the second phase, the robot would administer however many of the \( n \) allotted breaks remained. If the robot delivered \( m \) breaks during the first phase, it would have to deliver \( n - m \) breaks during the second phase. The breaks would be delivered along a fixed schedule in the second phase. The conditions are presented visually in Figure 2 (on the next page).
The fixed and random conditions would allow us to study two distinct aspects of the personalized condition. By comparing the personalized condition to the fixed condition, we’d be able to analyze the effects of variable break spacing. By comparing the personalized condition to the random condition, we’d be able to analyze the appropriateness of the technique used by the robot to determine when to administer breaks.

We considered adding a fourth condition that accumulates all break time and pushes it to the end of the tutoring session. This condition was intended to represent traditional teaching, where students are expected to finish all work before taking a break. The condition was also intended to be a control. We chose not to add it because it wasn’t comparable to the other conditions. The students in this condition would spend a substantial portion of time not practicing content with the robot before taking the study’s post-test. These students might begin to forget content as a result.

3.3.2. Personalized Condition

The addition of the personalized condition raises an important question: how should the robot determine when to administer breaks to students? At the highest level, students should be given breaks when their engagement levels drop. As a result, we need some measure of engagement. This ties back to Section 3.1.3, where I mentioned the need for engagement metrics that are more direct than learning outcomes. As noted in Section 3.1.3, timing and accuracy data are simple measures that can be easily attained and processed. Probabilistic models have been developed to translate timing data into engagement measures [15] and accuracy data into engagement measures [16].

**Measuring engagement using timing data.** In [15], Beck architects a probability function that takes a student’s response time as an input and outputs the probability that the student is disengaged. He uses data from a reading-tutoring program to both design and test his function. The reading-tutoring program specifically works with multiple-choice cloze questions (fill-in-the-blank questions used to test vocabulary).

Beck plots proportions of questions answered correctly against response times and reveals a logistic curve. Students who answered questions quickly tended to get them wrong, suggesting that
they were rushing or guessing, and vice versa. He proceeds to use item response theory, or IRT, which assumes that the probability of an individual answering a question correctly can be represented by a mathematical, usually logistic, function of individual and item parameters. Proficiency is a common individual parameter. The item parameters usually involve the question, for example, its difficulty and the probability of a student guessing the correct answer.

Beck designs a two-step process. He first uses an IRT-inspired logistic model to translate from response time (and a few other inputs) to the probability of a student answering a question correctly:

\[
P(\text{correct}) = c + \frac{a(1 - d) + d - c}{1 + e^{-a(t+sb(L_1+L_2))}}
\]

Equation 1.

where \(t\) is response time, \(L_1\) and \(L_2\) involve the length of the question, \(a\) determines the steepness (and thus distinguishing power) of the logistic curve, \(b\) is the question difficulty, \(c\) is the probability of a student guessing the correct answer, \(d\) is an upper bound on performance, and \(a\) and \(s\) are student-specific accuracy and speed parameters, respectively.

After obtaining the probability that a student will answer a question correctly, Beck performs a simple transformation to translate this probability into a disengagement probability:

\[
P(\text{disengaged}) = \frac{U - P(\text{correct})}{U - L}
\]

Equation 2.

where \(U\) is the upper bound of the logistic curve used in the calculation for \(P(\text{correct})\), and \(L\) is the lower bound. This transformation simply scales the value based on the range of the logistic curve and flips it, so to speak. A high \(P(\text{correct})\) translates to a low \(P(\text{disengaged})\).

Of the parameters in the Equation 1, only \(t, L_1,\) and \(L_2\) are entered in fresh for every question. The other parameters of the model have predetermined values that aren’t affected by subsequent user input. They were all calculated by running regressions on previous data from the reading-tutoring program. This includes the student-specific accuracy and speed parameters. To make this model more responsive, we could update the accuracy and speed parameters in real-time as students answer questions.

**Measuring engagement using accuracy data.** In [16], Schultz and Arroyo present a more dynamic model for measuring engagement. The model essentially merges two hidden Markov models, one which traces hidden knowledge states and one which traces hidden affect, or engagement states. For both models, the observed state is student performance on questions. The first model, the Bayesian Knowledge Tracing (BKT) model, takes the following inputs: the probability that a student already knows a skill, the probability that a student will learn the skill from one time-step to the next, the probability that a student who doesn’t know the skill guesses the correct answer, and the probability that a student who knows the skill makes a mistake and answers incorrectly. The second model, the HMM-IRT model, takes a static knowledge parameter as one of its inputs. This is a major
limitation of HMM-IRT; proficiency in a subject over the course of a tutoring session on that subject is likely going change, specifically increase. To address this, Schultz and Arroyo update the knowledge parameter of the HMM-IRT model at every time step with the knowledge state of a simultaneously running BKT model, chaining the two models together.

**Trigger condition.** Measuring engagement is just the first step in determining when to administer breaks. A trigger condition also needs to be selected. The simplest option is to set a static threshold. If engagement drops below this threshold, a break should be administered.

Szafir and Mutlu explore subtler approaches to this task [17]. Their study, which uses an EEG headset to measure engagement, sets two dynamic thresholds. The first compares the average slope of recent engagement levels to the average slope of engagement levels since the start of data collection. The second generates two least-squares-regression, or LSR, functions, one for the past 15-second interval and one for all data seen thus far. These two functions are then merged in a weighted average. The function for the past 15-second interval is given a much higher weight of 95%. If engagement levels drop below either of these two thresholds, it is inferred that engagement levels are about to drop substantially, and that action should be taken. To prevent excessive triggering, Szafir and Mutlu ignore a drop if one was recognized and acted upon in the past 15 seconds, a simple but important refinement.

**Using these models in our study.** Both the timing (Beck) and accuracy (Schultz and Arroyo) approaches to measuring engagement require parameters in a model to be set. For the timing approach in particular, a full plot of proportion of questions answered correctly vs. response time needs to be constructed. It’s from this plot that many of the parameters for the logistic function are selected. I initially thought that this would be possible given that an ITS study with fraction problems had already been run at Yale, but two problems arose. First, only limited timing data was collected during the previous study, and second, we eventually changed the problem domain from fractions to multiplication facts. This will be discussed in Section 3.3.3.

For both the timing and accuracy approaches, a preliminary dataset could be collected. We could visit a school, have students work on a set of problems, and record timing and accuracy data. This would help us set the parameters of the timing approach’s logistic function or the transition probabilities of the accuracy approach’s hidden Markov models.

Even if we aren’t able to use any of the models presented in this section, we can certainly seek inspiration from them when designing our own engagement metrics.

**3.3.3. Changing Problem Domain**

We eventually decided to shift from fraction problems to basic multiplication facts. This was done to make the duration of the study easier to control and to increase the granularity of the break provision. Because multiplication facts can be completed so quickly, we can place a time cap on them. We can also fit more problems into a tutoring session, increasing the number of points at which breaks can be initiated. Consider a tutoring session with only 10 problems and 3 breaks. The 3 breaks would have to be dispersed among 9 time slots. As a result, the variance across the three conditions’ placement of breaks would be minimal.
Another reason for switching the problem domain was to minimize the need for help. Yale’s previous ITS study allowed students to request hints. Allowing students to request hints during the current study would add yet another variable to the picture, making conclusions harder to process. (Hints and help-seeking behaviors lied at the heart of the previous study, so this wasn’t an issue.)

### 3.4. Final Experimental Design

At this point, I stopped addressing questions of experimental design and began implementing code for the study. The experimental design continued to evolve however. Before moving on to the implementation section of my report, I’ll briefly summarize the final state of the experimental design.

The conditions changed one final time:
- Condition 1; fixed: The robot administers breaks along a fixed schedule.
- Condition 2; personalized, reward: The robot administers breaks when it observes that a student is performing well.
- Condition 3; personalized, frustration (for lack of a better word): The robot administers breaks when it observes that a student is struggling or frustrated.

The last condition is particularly interesting. Could the condition encourage students to answer questions incorrectly? Or will it boost students’ morale as intended?

We also changed the problem domain again, from multiplication facts to order of operations problems. This refocuses the study on the teaching capabilities of ITSs. Multiplication facts are usually memorized, not learned. Order of operations problems hit a sweet spot. They’re complex enough to be learned, but they’re also simple enough to be completed quickly and understood even if one is new to the concept.

### 4. Implementation

The code that I’ve written is accessible at this GitHub URL: https://github.com/ScazLab/nao_tutoring. Note that the code for this study builds off of the code from Yale’s previous ITS study. Also note that multiple students have been pushing code to the repo. To see my changes in isolation, I recommend using the pull requests tab. (Navigate to Pull Requests → Closed.) I’ve grouped and documented my changes in pull requests as best as I can.

#### 4.1. ITS Architecture

The ITS used in the study has four components: a Nao robot, a tablet, a laptop, and a router. Nao is connected to the router via Ethernet, and the tablet and laptop are connected to the router via Wi-Fi. The laptop runs a Python script, `nao_server.py`, which controls Nao over the network. This same Python script receives messages from and sends messages to the tablet over the network. The laptop thus serves as an intermediary between Nao and the tablet, the two components of the ITS that the student directly interacts with. Figure 3 presents this information in a visual format.
Figure 3.

4.1.1. Message Passing

Message passing between the tablet’s Android app and nao_server.py on the laptop is central to the functioning of the ITS. When a button is pressed on the Android app, it (usually) sends a message to nao_server.py. Before sending the message, the app disables its buttons. This prevents students from proceeding forward while messages are being processed. Meanwhile, nao_server.py is constantly listening for messages. Whenever it receives a message, it parses the message and has Nao speak and move accordingly. Once Nao has finished speaking and moving, it sends a message back to the tablet. The tablet (usually) re-enables buttons upon receiving this message, and the session continues.

I made extensive use of this message-passing pipeline, especially when developing the tic-tac-toe break, which involves substantial interaction between the student and Nao, all mediated through the tablet.

4.2. Break Pipeline

For reference, I collaborated with Kevin Jiang on this task. I’ll note which subsets of the task were handled by Kevin.

The tablet application is an Android app consisting of Android activities. Two activities were implemented for the previous study: MainActivity and MathActivity. The former is used to configure the system. The second contains the actual tutoring interface. I introduced two new activities, one for the tic-tac-toe break, TicTacToeActivity, and one for the stretch break, StretchBreakActivity. The new activities are started from MathActivity. I had to modify MathActivity’s attributes to ensure that, whenever TicTacToeActivity or StretchBreakActivity finishes, the existing instance and not a new instance of MathActivity is
returned to. By default, Android creates a new instance and returns to it instead of the existing instance.

To trigger these breaks, I added a boolean field, `takeBreak`, to the `MathActivity` class. If this boolean is set to true, the next call to `MathActivity`'s `NextQuestion()` method will trigger a break instead of rendering a question, which is what the method normally does. The boolean is reset to false any time a break is triggered.

To trigger breaks in the fixed condition, a counter was added to `MathActivity`. Whenever this counter hits the specified threshold, `takeBreak` is set to true. Kevin implemented this.

Unlike the fixed condition, which is managed entirely within the Android app, the personalized condition involves communication between `nao_server.py` and the Android app. We decided to trigger these breaks from `nao_server.py` because the script is also responsible for logging, and there is significant overlap between the data that needs to be logged and the data that’s used to trigger breaks. To trigger these breaks, we make use of the message-passing pipeline described in Section 4.1.1. After receiving a message from the Android app indicating that a question has been completed, `nao_server.py` checks whether or not a break needs to be triggered. If this is the case, once Nao has finished speaking and moving, `nao_server.py` will send a special `BREAK` message to the Android app instead of its standard `c` message. Upon receiving this message, the Android app will set `takeBreak` to true. Kevin and I discussed this together, but Kevin implemented it.

4.3. Tic-Tac-Toe Break

I spent most of my time developing the break activities themselves. The interface for the tic-tac-toe break is presented in Figure 4.
4.3.1. Passing the TCPClient Object

To allow TicTacToeActivity to pass messages to nao_server.py as MathActivity does, I had to introduce a new Java interface, TCPClientOwner, to the app. Prior to the existence of this interface, only MathActivity could own the app’s TCPClient object. Now any class that implements the TCPClientOwner interface can do so. To ensure that ownership of the TCPClient object is restored to MathActivity when TicTacToeActivity or any other break activity finishes, I extended MathActivity’s onResume() method.

4.3.2. Message Passing

To make the user experience as smooth as possible, the tic-tac-toe break makes extensive use of message passing. The use of message passing is best explained with an example:

1. TicTacToeActivity sends a TICTACTOE-NAOTURN message to nao_server.py when the student selects a square. The tic-tac-toe board is disabled.
2. The script has Nao verbally acknowledge the student’s move. It then sends a TICTACTOE-NAOTURN message back to TicTacToeActivity. TicTacToeActivity’s TICTACTOE-NAOTURN handler doesn’t re-enable the board because it’s Nao’s turn. Instead, the handler calls the NaoTurn() method, which is used to select a square for Nao. An O is placed in the selected square.
3. The tablet checks whether or not Nao has won. Let’s assume for now that Nao hasn’t won. TicTacToeActivity then sends a TICTACTOE-STUDENTTURN message to nao_server.py.
4. The script has Nao let the student know that it’s his or her turn. It then sends a TICTACTOE-STUDENTTURN message back to TicTacToeActivity. TicTacToeActivity’s TICTACTOE-STUDENTTURN handler re-enables the board.

The sequence of messages is of course slightly different when a game ends because a player has won or it’s a tie.

Note that multiple games can be played. The game repeatedly restarts until a specified amount of time, stored in TicTacToeActivity's TIME_LIMIT constant, has passed. This functionality requires additional message passing and message types.

4.3.3. Nao’s Minimax Strategy

The tic-tac-toe break posed an interesting algorithmic challenge. We want Nao to play well enough that students aren’t bored, but we also don’t want to Nao to repeatedly win and make students feel down. To strike this balance, I implemented a minimax strategy with a variable depth parameter.

The minimax algorithm is a recursive algorithm that seeks to maximize a player’s score while recognizing that the player’s opponent seeks to do the same. With every subsequent level of recursion, the algorithm is essentially looking one move ahead. When the algorithm is allowed to recurse fully for a tic-tac-toe game, it can’t beat, only tied. On the other hand, if the algorithm’s recursion is capped, the algorithm doesn’t perform nearly as well. Say the depth is capped at 1 for example. The algorithm will only consider the current move. That is, it’ll make itself win when possible, but it won’t try to block its opponent from winning.

The depth parameter of Nao’s minimax strategy is initialized to 2. From here, every time the student wins, the parameter is increased by 1, thereby increasing Nao’s skill. Conversely, every time the student loses, the parameter is decreased by 1 (if possible), thereby decreasing Nao’s skill. In the case of a tie, the depth is left unchanged. After a few rounds of tic-tac-toe, Nao’s skill level should be calibrated with the student’s.

4.4. Stretch Break

The stretch break was relatively simple to implement. Unlike the tic-tac-toe break, it requires much less message passing. StretchBreakActivity sends one message to nao_server.py upon starting. nao_server.py then has Nao speak and move as necessary. Once nao_server.py has finished, it sends a STRETCHBREAK DONE message to StretchBreakActivity. StretchBreakActivity activates its “Return to tutoring session” button accordingly.

The most challenging piece of the stretch break was designing Nao’s motions. This involved a process of trial and error.

4.5. Additional Work

Although not the most interesting task, I’ve added a detailed README to the project’s GitHub repository describing how to get the project’s various components installed and running. This will hopefully be helpful for future contributors to the project. I’m also in the process of writing up
documentation that describes how to add new breaks to the project. This will allow others to quickly pick up where I'm leaving off.

5. Conclusion

5.1. Concluding Remarks

Even though I wasn’t able to see this study through to completion, I’ve learned a great deal about social robotics and experimental design. I want to thank Aditi Ramachandran for being a great mentor as well as Chien-Ming Huang and Professor Scassellati for all their help and advice!

5.2. References