An Inline Trainer to Support Teachers Learning to Use Technology

A Senior Honors Thesis

Presented to

The Faculty of the School of Arts and Sciences
Brandeis University

Undergraduate Program in Computer Science
Prof. Richard Alterman, Advisor

In partial fulfillment of the requirements for the degree of Bachelor of Science

By
Zachary Kolar

May 2018

Abstract

Technology has become an essential part of the education process. Teachers are asked to implement software and apps into their lessons. Teachers rarely have enough time, relevant training, or support to sufficiently accomplish this. Traditional training methods such as manuals or workshops take time away from teachers' other responsibilities and often present information outside of the context in which the technology will be used. Inline training offers a way to seamlessly incorporate training into users' normal workflow. An inline trainer takes advantage of natural interruptions as a user navigates a particular software and uses these interruptions as teachable moments. This reduces the amount of time dedicated to training, personalizes help to each individual user, and presents new knowledge in the context in which it will be used. In this study, an inline trainer was designed to help instructors learn to use Moodle, an open-source learning management system. The design was tested in an experiment consisting of ten subjects who were given a list of tasks to perform on Moodle and access to the trainer. Through observing and interviewing subjects, a model was created to represent users' workflows while completing tasks. This model was used to categorize three styles of use of the inline trainer: 1) trainer as backup; 2) trainer as guide; and 3) trainer as validator. Lists of the most important interface items, common uses of the trainer, and requested improvements were also extracted from the data. Based on these lists, recommendations are made for improvements to the trainer and future research. This project acted as a capstone for my Brandeis undergraduate experience because it gave me an opportunity to combine and use much of what I have learned in my computer science and education coursework to create a real tool to help teachers learn to use technology.

Acknowledgments

This project would not have been possible without the support of many people. For helping me to complete this journey, I would like to thank:

> Professor Richard Alterman, my advisor throughout this process. After completing three classes, two independent studies, an experiential learning practicum, and now this honors thesis under his tuition and guidance, I cannot overstate how thankful I am for the impact he has made on me as a computer scientist, designer, and educator.
>
> Maria Altebarmakian and Hannah Moser, who provided input, advice, and laughter along the way.
>
> Dr. Brian Krisler, for providing support and insight.
>
> Dr. Karen Muncaster, Chad Bergeron, and Adam Lipkin, for their enthusiasm and support to see this project through.
>
> Dr. Lisa Rourke and the University Writing Seminar instructors, for helping me design and test the inline trainer plugin.
>
> Professor Jennifer Cleary, Eileen Kell, Professor Marya Levenson, Professor Mitra Shavarini, and Professor Derron Wallace, for encouraging me and helping me pursue my interests in education, technology, and the overlap between them.

Finally, I would like to thank my friends and family, who always have always provided me with endless support and encouragement throughout this project, my Brandeis career, and my whole life.

Contents

List of Figures

List of Tables

1 Introduction

Preparing teachers to use technology is a is a multifaceted issue. In addition the challenges faced by the general population when it comes to learning to use technology, teachers face a unique set of challenges such as the lack of time to to learn and practice technology as well as insufficient access to technical support (Becta, 2004; Peck et al., 2011; Toprakci, 2006; Dexter et al., 2002; Kirkwood et al., 2000; Inan and Lowther, 2010). As technology becomes more prevalent in the classroom, teachers need a simple way gain technical proficiency in a way that does not overlap with their other duties. An inline trainer (Krisler, 2014) could help teachers learn to use a particular software as they use it, without requiring additional outside training. Such a trainer would act as a reference on the edge of the screen, providing interactive instructions to complete actions as necessary.

In this paper, I will review the literature relevant to learning, technology training, teachers and technology, and inline trainers. I will then describe the design of an inline trainer created as a plugin to help instructors learn how to use Moodle[1] to manage their courses. I will detail a pilot study involving users completing Moodle tasks with the assistance of the trainer. The purpose of the study was to:

i. Categorize different styles of use of the inline trainer
ii. Create a model to represent the ways in which users incorporate the inline trainer into their workflow as they navigate the Moodle interface
iii. Identify the features and uses of the trainer the users perceive as most helpful
iv. Generate a list of improvements for future iterations of the inline trainer

Based on the data collected in the study, I will present a model representing the ways in which subjects completed tasks within Moodle. Subjects' use of the trainer within this model was classified into three distinct styles: 1) trainer as backup; 2) trainer as guide; 3) trainer as validator. Additionally, I will highlight important interface features, uses of the trainer, and requested improvements. Based on the data gathered from the pilot study, I will recommend changes to the trainer for further research and use as an instructional tool.

1.1 Overview of the process

The goal was to develop an approach to training teachers to use technology. I chose to focus on Moodle, as it is the learning management system (LMS) at Brandeis and is therefore familiar to faculty, staff, and students. Moodle is a powerful LMS, but complex and often difficult to navigate. Because of this, many instructors use only its most basic features. To help instructors learn to use more features of Moodle, I designed and developed an inline trainer. The idea of an inline trainer is to use naturally-occurring interruptions in the use of software as opportunities to learn more about the software. This project enabled me to practice the things that I learned during my undergraduate education in both computer science and education.

1.1.1 HCI and Software Development

After reviewing Dr. Brian Krisler's 2014 previous work with inline trainers, I created a proof-of-concept plugin to see if such an inline trainer was technically viable in the context of Moodle. After determining such a plugin was possible, I began to collect data via informal surveys to determine the types of actions that should be included and the taxonomy under which they should be classified. After collecting this background data, I created a prototype of the inline trainer outside of Moodle. I obtained feedback on the design and functionality of the prototype through informal demonstrations and trials.

Based on the feedback I received for the prototype, I created the earliest iteration of the current

[1] Moodle is an online, open-source learning management system

Moodle plugin. I conducted another round of surveys of faculty at Brandeis University to determine which types of Moodle actions they knew and did not know, as well as gauge which types of actions they would be interested in learning. I met with instructors to discuss their workflow in Moodle to develop use cases. I consulted with members of the Brandeis technology support department to talk about the feasibility of installing and maintaining such a plugin on the instance of Moodle at Brandeis, known as the Learning and Teaching Technology Environment (LATTE). Using this feedback, I continued to develop the plugin.

Further discussion of the various stages of development as well as links to the source code repositories can be found in Appendix A.

1.1.2 Experimental design, data collection & analysis

The pilot study involved a group of subjects who were given tasks to complete on Moodle. I created the tasks based on the use cases from the development phase. I chose a number of tasks of varying levels of difficulty and familiarity[2]. The experiments were conducted in two rounds, with changes made to the trainer interface between the two rounds based on observations and feedback from the first round. The first round consisted of two sessions per subject with one day in between the sessions; the second round consisted of one session per subject. Each session had a different list of tasks for subjects to complete.

I developed a second Moodle plugin to help administer the experiments. The plugin allowed me to quickly create user accounts and test courses that were pre-populated to resemble the Moodle pages of existing Brandeis course. Further discussion of the experiment plugin as well as a link to the source code repository can be found in Appendix B.

Before subjects completed their assigned tasks, they responded to a survey of demographic information and their experience and comfort level using technology. As they completed the tasks, they were instructed to talk aloud. The computer screen and subjects' voices were recorded for analysis.

The videos were coded using Atlas.TI[3]. The built-in analysis features in Atlas.TI are limited, so I created a web-based tool to parse the XML Atlas.TI project files. I used this to compare and analyze the codes applied to the video files. Further discussion of the analysis tool as well as a link to the source code repository can be found in Appendix C.

Based on the analysis, I created a model of subjects' workflows as they completed the tasks. From this model, I highlighted three distinct styles in which the trainer was utilized. I also created lists of important features and uses of the trainer, as well as suggested improvements.

[2] I determined the functionality with which faculty were most and least familiar through informal surveys

[3] ATLAS.ti: The Qualitative Data Analysis & Research Software (`https://atlasti.com/`)

2 Review of Literature

2.1 Technology Training and Learning

2.1.1 Cognitive Skill Acquisition

VanLehn (1996) defines cognitive skills as skills necessary for any problem that relies primarily on specific knowledge, rather than physical abilities, to solve. Cognitive skill acquisition occurs in three phases:

Early: The learner looks to gain domain knowledge before trying to apply it. This phase generally involves information-gathering through reading and discussion.

Intermediate: The learner corrects flaws in domain knowledge and begins to apply the skill to real problems. During this phase, the learner acquires skills by retrieving prior knowledge, mapping it to the current problem, and then generalizing it to solve future problems.

Late: The learner practices the skill to gain speed and accuracy. The time the learner takes to perform the task decreases proportionately to the number of times it has already been performed, raised to some power.

The distinction between each of these phases is not always clear in practice. The phases may also overlap. Nevertheless, the three phases provide a general framework with which to study cognitive skill acquisition.

2.1.2 Distributed Cognition

Perkins (1997) suggests that human cognition extends beyond the person and into the surrounding environment. This person plus surround, or *person-plus*, approach includes any tools a person may use to help store and retrieve knowledge, such as a book, notebook, or other artifact. This provides a more power cognitive system than *person-solo*, which is just the knowledge contained in the mind of a person. Within a person-plus system, executive function is often distributed in many ways. If a person is learning a new skill, he or she may relinquish executive control to the surround. This may be in the form of reading a book, following instructions given by an instructor, or working through a worksheet. Executive function is gradually taken back by the learner as the knowledge is transferred.

2.1.3 Cognitive Artifacts

Norman (1991) defines a cognitive artifact as "an artificial device designed to maintain, display, or operate upon information in order to serve a representational function" (p. 17). Although artifacts enhance performance, they do not amplify abilities. They modify the task that must be performed in such a way that a person can complete them more easily. They change tasks in three main ways (Norman, 1991, p. 22):

> Distribute the actions across time (precomputation);
> Distribute the actions across people (distributed cognition);
> Change the actions required of the individuals doing the activity.

For example, consider the process of grocery shopping. Without any artifacts, the task takes place primarily at the grocery store. The person must mentally track which items have been purchased and which are still needed. Introducing an artifact, such as a list, changes the task. The cognitive work is front-loaded during the construction of the list, which takes place before the shopping trip. While the person is shopping, they no longer need to rely on their memory – her or she can refer to the list and cross items off as they are purchased (Norman, 1991).

2.1.4 Technology training

As the role of technology in jobs continues to grow, more attention is given to the the ways in which training is structured. Some technology training focuses on features of the software rather than its usage. Through this model, users may be able to explain *what* components of the software can do, but not *how* use them. Because trainees often do not hold a role whose primary function is to use technology, an approach that focuses on the *how* aspects is necessary for effective training Bostrom et al. (1986).

End-user training (EUT) is one approach to technology training that aims for two outcomes: 1) trainees gain a basic understanding of the system and 2) trainees have high motivation to use the system. In order to gain a basic understanding, trainees must be able to create a mental model of the system. This mental model represents the user's understanding of the system at hand. The basis of the mental model can originate externally from the user, but it must be internalized through exploration and explanation for the training to be effective (Bostrom et al., 1986). Laoledchai et al. (2008) propose an expanded version of EUT that also takes attitude into consideration. Even if trainees gain new skills or abilities during training, they are unlikely to transfer these into their work if they believe the training to be ineffective (Laoledchai et al., 2008).

2.2 Teachers and technology

2.2.1 Problems with existing technology training

Lack of proper technology training and/or ongoing support are two of the perceived largest barriers that discourage teachers from utilizing technology (Becta, 2004; Peck et al., 2011; Toprakci, 2006; Dexter et al., 2002; Kirkwood et al., 2000; Inan and Lowther, 2010). Inan and Lowther (2010) studied eight factors hypothesized to affect teachers' use of technology: 1) age, 2) years of teaching, 3) computer proficiency, 4) computer availability, 5) teachers' beliefs, 6) teachers' readiness, 7) overall support, and 8) technical support. Out of these factors, teachers' readiness was found to have the strongest direct effect on technology integration. Teachers' readiness was most strongly affected by their computer proficiency, followed by overall support and technical support. In other studies, teachers expressed frustration that they were given short (if any) trainings and expected to learn the rest on their own time (Becta, 2004; Peck et al., 2011). As part of a case study in a U.S. high school where digital tablets were given to all staff with limited training, Peck et al. (2011) found that some teachers abandoned the devices all together after becoming frustrated at the lack of training they received and the amount of time learning to use the tablets independently would take out of their instructional time.

Technology training is often not delivered in a way that is relevant to teachers (Granger et al., 2002; McKenzie, 2001; Kirkwood et al., 2000). McKenzie (2001) notes that training often lacks differentiation for personal preferences or technological abilities. Kirkwood et al. (2000) found that traditional teacher technology training often focuses on just the basics of using a particular technology, which are quickly forgotten because other aspects of teaching take priority in teachers' minds. Additionally, technology instruction is often provided outside of the context of the classroom (Granger et al., 2002; McKenzie, 2001). Technology training tends to focus primarily on the functionality of various tools and not ways in which the tools can be implemented in the classroom. Without contextualizing the use of technology into teaching practice, many teachers are left to wonder whether the technology is actually a valuable tool (McKenzie, 2001).

Even when given sufficient training, teachers often lack access to ongoing technological support as they implement technology in their classes. A national survey of teachers found that access to quality technology support was associated with increasing use of technology (Dexter et al., 2002). It was also also found that quality technology support was correlated with more frequency and variety of uses of technology over time. Thorsteinsson and Niculescu (2012) concluded that teachers who are not comfortable with technology rely on support and often must go out of their way to get it. Meanwhile, teachers who are more familiar with technology help their peers and, which interrupts them from completing their own work. Teachers have very limited time to devote to learning and incorporating technology Kirkwood et al. (2000); Becta (2004); Thorsteinsson and Niculescu (2012). Kirkwood et al. (2000) observed that a voluntary training program for teachers is difficult to justify, because it requires them to give up time that could be otherwise spent completing other tasks. Becta (2004) noted that in general, schools do not have enough time to provide meaningful technology training for teachers. Thorsteinsson and Niculescu (2012) found that even looking at technology-related help materials provided to teachers could be too time-consuming and therefore ruled out as an option as problems arose.

2.2.2 Recommended best practices for technology training

Zhao and Cziko (2001) propose Perceptual Control Theory (PCT) as a framework to understand teachers' use of technology. They suggest that human behavior is a result of trying to achieve internal goals. In order for someone to behave a certain way, they must internalize the end result as a goal towards which to strive. Because of this, PCT requires that three conditions are met in order for teachers to use technology (Zhao and Cziko, 2001, p. 6):

1. The teacher must believe that technology can more effectively meet a higher-level goal than what has been used.
2. The teacher must believe that using technology will not cause disturbances to other higher-level goals that the he or she thinks are more important than the one being maintained.
3. The teacher must believe that he or she has or will have sufficient ability and resources to use technology.

Kirkwood et al. (2000) recommend differentiating training to meet individual teachers' needs based on their level of use and understanding of technology. Teachers with less technology experience need more scaffolding to fully engage with technology.

Burns and Dimock (2007) propose a *5J approach* to technology training (p. 23):

Job-related: Focused on classroom (i.e., instructional and curricular) needs

Just enough: Emphasizing increased comfort, not proficiency, with computers

Just in time: Providing teachers with skills as needed and focused on using only the tools they have at their immediate disposal

Just in case: Encouraging teachers to plan sufficiently in the event of a computer malfunction

Just try it: Applying enough pressure and support to impel teachers to use computers in their classrooms

"Just enough" and "just in time" address the previously-discussed issue of providing context during technology training. These two Js ensure that training only encompasses what teachers need to know in order to leverage technology as pedagogical tools. It keeps the training brief and relevant to their classes. Granger et al. (2002) found that teachers reported that just in time learning is the most effective way for them to learn technology. This informal approach gives teachers a tangible experience during learning which makes it easy to connect to their practice. It also gives them more autonomy over their learning and helps them build the skills necessary to keep learning as they need Granger et al. (2002).

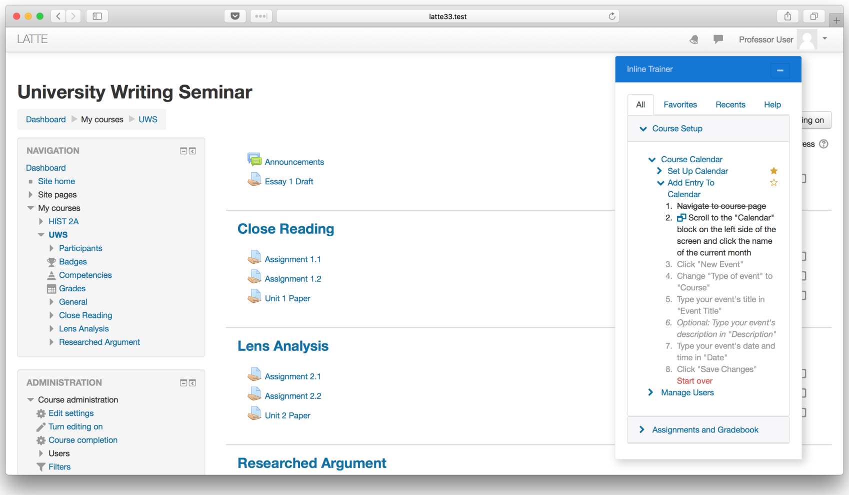
2.3 Inline trainer

An inline trainer is a tool designed to help people learn to use software by using naturally occurring interruptions to introduce new application knowledge (Krisler, 2014). The trainer is a form of scaffolding, which refers to a technique where support is provided to novice users and gradually removed as their proficiency builds (Wood et al., 1976). During task interruptions that are caused by a lack of application knowledge, the user's exploration of the functionality of the application can be mediated by the use of the inline trainer. In this manner, exploration of new functionality is scaffolded by the trainer. Over time, the user learns to perform a particular task and the trainer becomes less necessary for the completion of that task.

There are many sources of interruptions while using software, but one common type is an inquiry interruption (Jin and Dabbish, 2009). An inquiry interruption occurs if a user is unable to continue with the current task because they do not have enough knowledge about the application. In order to recover from this interruption, the user must redirect their attention away from the task. Bødker (1995) categorizes redirects as either breakdowns or focus shifts. Breakdowns are caused externally from the user, such as when the tool behaves in an unexpected way. Focus shifts occur when the user deliberately diverts his or her attention from the task, such as to explore the tool to determine how to proceed (Krisler and Alterman, 2008).

In EUT model discussed earlier, Bostrom et al. (1986) describes the process through which mental models are formed. The two ways to start this process are mapping via usage and mapping via training (i.e. using an external representations of the functionality of the system). The inline trainer allows the user to go through both of these routes. The user is free to explore the interface and build up his or her mental model through usage and completion of tasks. When an interruption occurs due to lack of knowledge about how to use the application, the user can seamlessly transition to the representation of the system contained in the trainer to further develop his or her mental model.

3 Design of the Inline Trainer



1. ~~Navigate to course page~~
2. Scroll to the "Calendar" block on the left side of the screen and click the name of the current month
3. Click "New Event"
4. Change "Type of event" to "Course"
5. Type your event's title in "Event Title"
6. *Optional: Type your event's description in "Description"*
7. Type your event's date and time in "Date"
8. Click "Save Changes"

Figure 3.1: When not collapsed, the trainer sits on the edge of the screen.

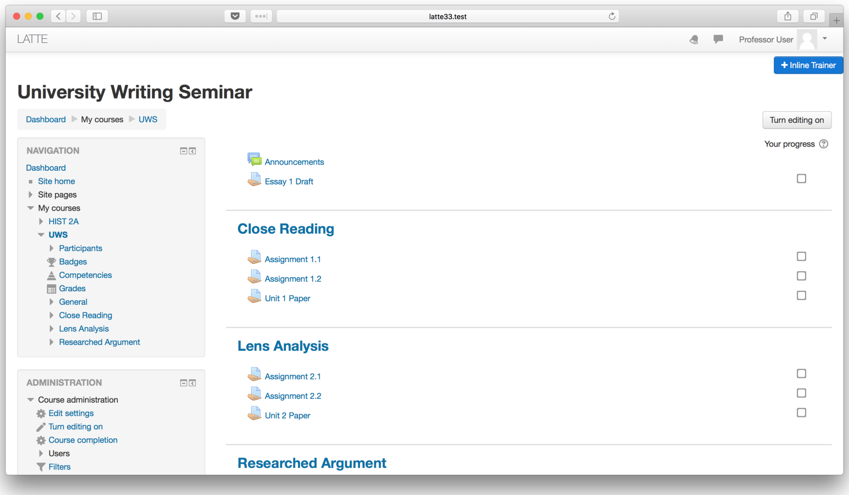
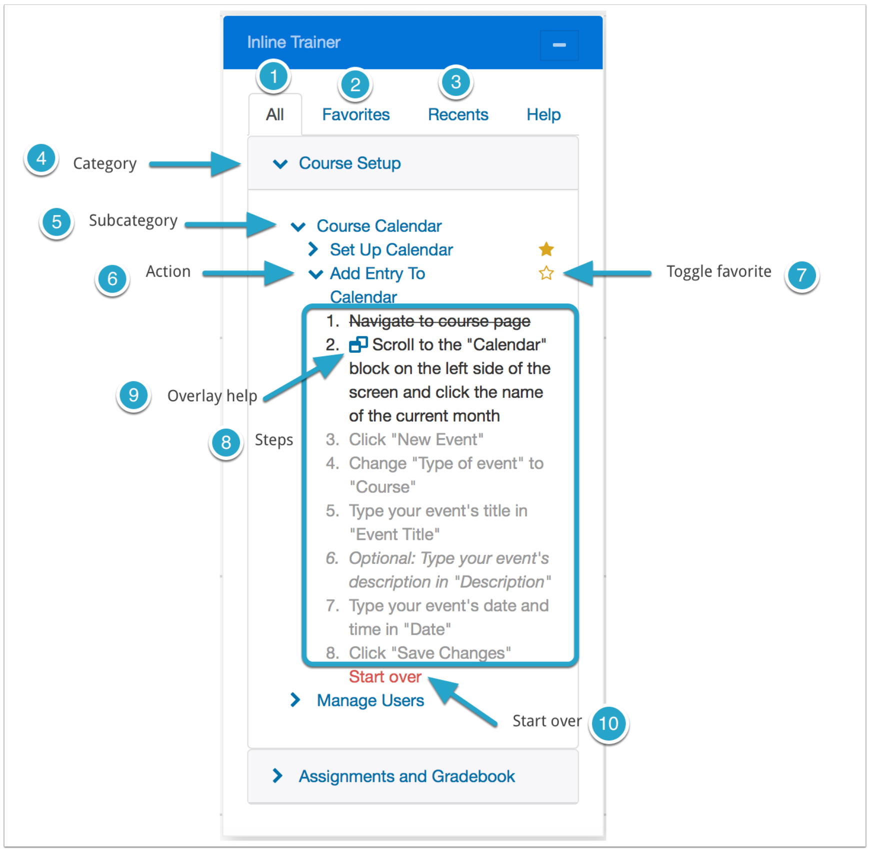


Figure 3.2: When the trainer is collapsed, it displays as a small blue button in the upper-right corner of the screen. Clicking the button un-collapses the trainer.

The inline trainer was developed as a plugin for Moodle, based on prior research by Krisler (2014). It was designed to sit on the side of the screen to provide easy access when users need it and be unobtrusive when they do not (Figure 3.1). Users can move the trainer to any position on the screen by clicking and dragging the blue title bar at the top. Because the trainer is directly embedded in Moodle, users can access information to help them accomplish tasks while minimizing the scale of the interruption necessary to do so Krisler (2014). The trainer can be collapsed into a small button when it is necessary to see the whole screen (Figure 3.2).



1. ~~Navigate to course page~~
2. Scroll to the "Calendar" block on the left side of the screen and click the name of the current month
3. Click "New Event"
4. Change "Type of event" to "Course"
5. Type your event's title in "Event Title"
6. *Optional: Type your event's description in "Description"*
7. Type your event's date and time in "Date"
8. Click "Save Changes"

Figure 3.3: Anatomy of the inline trainer

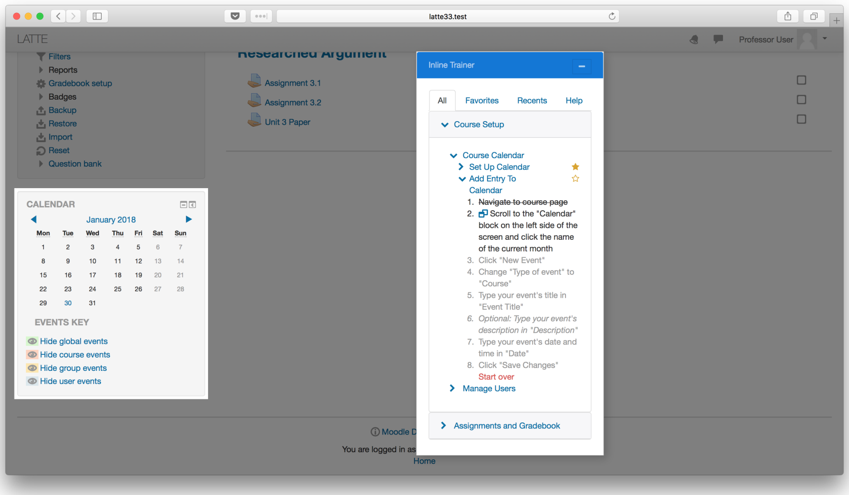
The trainer uses a tabbed viewing system to help users easily keep track of and find actions. The "All" tab (1 in Figure 3.3) provides a list of all the actions. The "Favorites" tab (2 in Figure 3.3) provides a list of actions users have identified as their favorites, allowing them to quickly pick out these actions from the list. The "Recents" tab (3 in Figure 3.3) shows a list of the five most recently used actions.

In the "All" tab, actions (6 in Figure 3.3) are organized into a taxonomy of categories (4 in Figure 3.3) and subcategories (5 in Figure 3.3). Users can click the names of categories, subcategories, and actions to open and close them. Clicking the star icon next to action names (7 in Figure 3.3) toggles the action as a favorite. An action is in the "favorites" list when the star is solid gold.

When an action is open, a list of steps (8 in Figure 3.3) is displayed. Completed steps are crossed out and future steps are written in gray text to help the user identify the current step that needs to be completed.

If the user is unable to figure out how to complete the current step in the Moodle interface, he can click the "overlay help" icon (9 in Figure 3.3) to display an overlay on the screen that identifies the item required to complete the step (Figure 3.4). If the trainer is blocking the item necessary to complete the action, it will slide out of the way when the "overlay help" button is clicked. Similarly, if the item is not within the view of the browser window, the trainer will scroll up or down on the page until the item is in view. The scrolling is animated to help the user understand what is happening and identify where the item is on the page relative to their original position.

The trainer automatically keeps track of step completion as the user navigates Moodle, even



1. ~~Navigate to course page~~
2. Scroll to the "Calendar" block on the left side of the screen and click the name of the current month
3. Click "New Event"
4. Change "Type of event" to "Course"
5. Type your event's title in "Event Title"
6. *Optional: Type your event's description in "Description"*
7. Type your event's date and time in "Date"
8. Click "Save Changes"

Start over

Figure 3.4: The trainer provides overlays to help users identify the target element of the Moodle interface in a given step

---

when the trainer is collapsed. Because of this, a user can start an action without using the trainer. If he gets stuck, he can open the trainer, find the action he is trying to complete, and see exactly what he needs to do next. The trainer also watches for steps to be undone in case the user makes a mistake or gets lost in the process of completing the action. The trainer is capable of keeping track of multiple step completions or un-completions at once. For example, if a user is on step 6 of an action and accidentally navigates to a page that undoes steps 2-5, the trainer will automatically set the current step to 2 and walk the user through the steps that have been undone.

Each action has a "Start over" button (10 in Figure 3.3) that resets the action to the first step in case the user needs to repeat an action or becomes confused and wishes to start over.

Interactions that affect the display of the trainer (e.g. collapsing and repositioning the trainer; opening and closing categories, subcategories, and steps) are automatically stored and retrieved as the user navigates between pages on Moodle so that the trainer display remains consistent.

4 Pilot Study

4.1 Motivation

Krisler and Alterman (2018) established inline training as a way to promote a deeper understanding of software applications built into users' workflows. The Moodle inline trainer was motivated by a need to optimize the technology training provided to teachers. Such a trainer could save time and provide teachers with support, which are two of the largest barriers to teachers' use of technology (Becta, 2004; Peck et al., 2011; Toprakci, 2006; Dexter et al., 2002; Kirkwood et al., 2000; Inan and Lowther, 2010).

4.2 Research Questions

The research seeks to:

i. Create a model to represent the ways in which users incorporate the inline trainer into their workflow as they navigate the Moodle interface
ii. Categorize different styles of use of the inline trainer
iii. Identify the features and uses of the trainer the users perceive as most helpful
iv. Generate a list of improvements for future iterations of the inline trainer

4.3 Experimental Design

The experimental design consisted of a series of tasks, each designed to replicate the workflow of a college-level instructor setting up a course page in Moodle, an open-source Learning Management System (LMS) used by Brandeis University and many other higher education institutions (Costello, 2013). The tasks fell into three main categories based on functionality in Moodle: course calendar, assignments, and grade book.

Two sets of user observations were conducted. The first round of observations had eight participants and the second round had two. Changes were made to the trainer interface between the two rounds in response to user feedback and use of the trainer in the first set of observations. These changes are detailed in Appendix D.

The first round of observation included sessions on two separate days with one day in between (Session 1.1 and Session 1.2). The first day included four tasks and the second day included two. Prior to starting the tasks on the first day, subjects were provided with a printed and computer-based copy of Moodle documentation. They were also shown a video overviewing the trainer interface and functionality. Finally, they were given a sample task to complete using the trainer as a guide to practice using the interface. Subjects were told they could refer to the documentation and trainer as necessary throughout the rest of the tasks, however neither was required. On the second day, subjects were given access to and reminded of the documentation and trainer. Once again, they were told they could use either but were not required to do so.

The second round of observation took place with a single session for each subject (Session 2.1). The subject was given three tasks to complete. They were given the same instructions, documentation, and video as the subjects on the first day of the first round of observations.

4.3.1 Subjects

All participants in the study were affiliated with Brandeis as faculty or students. There were 10 subjects: 2 instructors, 1 department administrator, 1 graduate student, and 6 undergraduate students. All had some degree of experience using LATTE[4] as an instructor, teaching assistant, student, or some combination of these roles. Subjects were not provided with any compensation.

[4]Learning and Teaching Technology Environment, the instance of Moodle used at Brandeis University

Prior to the pilot study, subjects were asked to complete a survey of demographic information and experience using technology. One question asked subjects to evaluate their level of technological literacy by choosing the statement with which they identified the most. The statements and number of subjects who identified with each is listed in Table 4.1. A full list of survey questions can be found in Appendix E.

Statement	Number of subjects
I am excellent with technology; I can help others use it	2
I'm very comfortable with technology	4
I can use technology	3
I can use basic technology, but in a limited way	1
I'm uncomfortable with technology	0

Table 4.1: Subjects were asked to select the statement with which they agreed the most prior to completing the tasks on their first session.

4.3.2 Tasks

Figures 4.1, 4.2, and 4.3 show the list of tasks users were asked to perform. Each task specified the necessary content for completion (e.g. title, description, due date, and point value of assignment). The tasks were chosen to represent a variety of features in Moodle. They were presented in order of increasing difficulty, which was determined by several factors. The first factor was the number of steps required to complete the action. Actions that required relatively fewer steps were considered easier. The second factor was the depth of the mental model of Moodle's interface required to complete the tasks. Simpler tasks do not require extensive knowledge of the way Moodle operates. For example, creating an assignment only requires general knowledge of creating a new item in a course. After the process to create a new item is initiated, the interface provides all of the information necessary to complete the task. Complex tasks require more background knowledge about the way Moodle processes internal objects. For example, setting up the grade book requires knowledge of Moodle's grading structure, types of grade aggregation, and the levels at which these types can be specified. This knowledge is not presented directly in the interface; it requires exploration or external guidance. The third and final factor was the likelihood that the user had completed a given task (or similar task) previously. This was determined by surveying previous Moodle users and consulting the team who runs Moodle at Brandeis.

1.0 Sample task - set up course calendar
1.1 Create a specified calendar entry
1.2 Edit the created calendar entry
1.3 Create two specified graded assignments (1.3.1 and 1.3.2)
1.4 Create two specified ungraded assignments (1.4.1 and 1.4.2)

Figure 4.1: Tasks given to users during Session 1.1

4.3.3 Informational materials

The manual given to subjects was provided as a printed document as well as a PDF on the computer used to complete the tasks. It was composed of documentation published on the Brandeis website

2.1 Create two specified graded assignments (2.1.1 and 2.1.2)
2.2 Set up the the course grade book based on specified syllabus

Figure 4.2: Tasks given to users during Session 1.2

3.1 Create a specified calendar entry
3.2 Create two specified graded assignments (3.2.1 and 3.2.2)
3.3 Set up the course grade book based on specified syllabus

Figure 4.3: Tasks given to users during Session 2.1

that was relevant to the tasks the subjects were asked to perform.

The video that was played for users overviewed the functionality of the trainer. It included the manipulation of the trainer (moving/hiding/showing, hiding/showing categories, subcategories, and tasks), the layout of steps required to complete a task, the behavior of the trainer as the user completes tasks, and the overlay button. The video lasted approximately 80 seconds.

4.3.4 Apparatus

For the experiments, a customized instance of Moodle 3.3, referred to as Moodle Optimized for Conducting Human Analysis (MOCHA), was deployed. MOCHA was used exclusively for the purposes of the experiment and contained no real course data. It was accessible publicly via the internet, with security settings enabled so that no data were visible without logging in. Only the researcher had credentials to access the site.

MOCHA was set up with the built-in Clean theme to provide subjects with an interface very similar to that of LATTE, which also uses the Clean theme lightly customized through CSS. Because of this, the MOCHA and LATTE interfaces had differing colors but otherwise nearly identical layouts.

The MOCHA environment had two custom plugins, both developed by the author for this research. The first plugin added the inline trainer interface and controls necessary to display it. The second plugin automated the process of creating user accounts and courses for the subjects in the experiment. The technical details and links to the source code for these two plugins can be found in the appendices. The courses were modeled after a set of writing courses offered at Brandeis University to provide familiarity to participants. The courses were set up to use the "topics" display mode and pre-populated with three topics: "Close Read," "Lens Analysis," and "Research Synthesis." They were not populated with any other content.

Each participant had a separate user account given "teacher" privileges for unique courses on MOCHA. No participants had access to the same course. Usernames and passwords were generated by the researcher. The researcher signed into the corresponding account prior to starting each session, eliminating the need to share login credentials with the participants. Participants who performed tasks over two sessions were given separate courses for each session. This gave them the opportunity to start without clutter or potential mistakes from previous sessions while preserving the data they created.

The experiment was proctored by the researcher. Subjects performed the tasks on a 15" late 2013 MacBook Pro running macOS 10.13 High Sierra with a high-speed wireless internet connection. The taskbar had Finder, Google Chrome, Quicktime, and shortcuts to the PDF version of the Moodle documentation and trainer introductory video. Subjects accessed MOCHA using the web browser Google Chrome. They were logged in through an "incognito window" in Chrome to ensure their browsing session would be destroyed upon completion and closing of the window. During the experiment, Quicktime ran in the background (with the subject's consent) to record the screen and audio of the experiment session. The researcher sat adjacent to the subject throughout the session to observe and note use of the trainer for follow up questions.

4.3.5 Procedure

Each subject was required to sign a consent form prior to the start of the experiment. The subject then filled out a web-based demographic survey that included information about experience using specific functionalities in LATTE and general comfort level in using technology. After the survey, the researcher gave the subject the printed documentation, showed the subject the PDF version, and played the video about the inline trainer. Screen and audio recording began after the video. The use of video recording was included in the consent form and explicitly explained to the subject. The subject was given a sample task of setting up the course calendar and instructed to use the trainer to become acquainted with it. After completion of the sample task, the subject was given a handout with the rest of the tasks and the following instructions:

- They are a professor of a writing course setting up their course page at the beginning of the semester, which requires them to complete the following tasks
- They may use the written tutorial or inline trainer as necessary, but are not required to use either
- If they are unable to complete any of the tasks, they may skip them
- Getting stuck on a task is a reflection of the design of the interface, not their technical abilities
- They should talk aloud with their thought process as they complete their tasks

After the subjects completed the tasks, the researcher followed up with a list of questions (Figures 4.4, 4.5, 4.6). QuickTime continued to record to capture the subjects' answers.

In the first round of experiments, the subjects returned two days after the first session for a second round of tasks. They were given the same set of instructions as the first session with a new list of tasks (Figure 4.2). After the tasks were completed, participants were asked a different set of questions (Figure 4.5).

1.1 Was the trainer helpful? If so, how?
1.2 Would you incorporate the trainer into your workflow if you had to perform instructor tasks in the real version of LATTE?
1.3 How would you compare using the trainer for the calendar tasks vs. the assignment tasks?
1.4 What would you change about the trainer?

Figure 4.4: Questions asked after Session 1.1

2.1 Was the trainer helpful? If so, how?
2.2 Would you incorporate the trainer into your workflow if you had to perform instructor tasks in the real version of LATTE?
2.3 How would you compare using the trainer for the assignment tasks vs. using it for the grade book setup?
2.4 If the subject did or did not use the trainer in unexpected ways throughout either session, the experimenter followed up with specific questions.

Figure 4.5: Questions asked after Session 1.2

3.1 Was the trainer helpful? If so, how?
3.2 Would you incorporate the trainer into your workflow if you had to perform instructor tasks in the real version of LATTE?
3.3 How would you compare using the trainer for the assignment tasks vs. using it for the grade book setup?
3.4 What would you change about the trainer?
3.5 If the subject did or did not use the trainer in unexpected ways throughout the session, the experimenter followed up with specific questions.

Figure 4.6: Questions asked after Session 2.1

4.3.6 Data collection and analysis

Data were collected by recording the screen and voice of the participant while they were completing the tasks and answering the questions. After each session was complete, the video was saved and uploaded to a private Dropbox folder. Videos were imported to Atlas.TI, software to help with qualitative data analysis. In Atlas.TI, they were coded according to types of tasks, styles of use, and problems encountered.

Based on the video and the subjects' spoken comments, each task was transcribed to help trace through the operations. Example transcriptions are provided in Table 4.2. The transcriptions were used to help build and validate a model of users' workflows with the trainer and characterize the different ways in which the trainer was used. The model and characterizations are discussed in the Results section.

Subject	Task	Description
kqmp	1.1	Opens "add entry to calendar" in trainer. Reads "scroll to the calendar block…" step. Scrolls down and finds calendar block. Notes that event is on "21st of March" and clicks title of month. Reads "click new event" step. Scrolls down page to look for button. Scrolls back up, moves trainer over, reveals button blocked by trainer. Clicks button. Fills in title. Fills in date. Clarifies meaning of tasks on handout. Clicks save button.
kqmp	1.2	Moves mouse over gear icon, does not click. Opens "edit calendar entry" action in trainer. Reads "click the gear icon" step. Clicks gear icon. Scrolls down to duration section of form and specifies duration. Clicks save button.

Table 4.2: To help analyze the data more easily, each task was transcribed for each user. Transcriptions included interactions with the page and anything the user said aloud as they completed the task.

5 Results

The interview questions (Figures 4.4, 4.5, and 4.6) asked users whether they found the trainer useful and if they could envision themselves incorporating the trainer into their workflow if fulfilling the role of an instructor in real life. Subjects unanimously answered yes to both. This section will look at a model of the ways in which the trainer was used, specific styles in which subjects used the trainer, and specific perceptions users had about using the trainer.

5.1 Model of Trainer Use

A model of the process through which users completed the tasks and a sample iteration through the model are shown in Figure 5.1. The rectangles represent various states in which subjects found themselves as they completed tasks. Each state is labeled with a question the subject implicitly or explicitly used to exit the state. The transitions between the states are labeled with the answer to the state question.

In the model, the tasks correspond to the lists of tasks described in the section. An operation is defined as an atomic step towards completing an action, such as clicking a button or filling in a form field. A session is defined a period of time in which subjects completed a distinct set of tasks. The entry point to the model is the session to allow for transitions to and from tasks, even after completion.

Two questions acted as turning points for the use or non-use of the trainer: “Do I know what to do?” and “Do I know how to do it?” Although these two questions are similar, there were instances where the subject knew *what* to do next (e.g. “I need to click *something* to bring up the edit page for this calendar entry”) but not *how* to accomplish it (e.g. “I do not know what to click to make this happen”). Subjects who entered the “Do I know what to do?” state and initially followed the “no” path tended to discover *what* to do and *how* to do it simultaneously, either by searching the interface and finding an element they deemed helpful or by checking the trainer.

The model is written from the perspective of the user. Therefore, “knowing” what to do next or how to do it means the subject has decided how to proceed. It does not imply that the operation will advance them further in the completion of the current task.

5.1.1 Three different styles of usage

The model was created to encompass almost any path subjects took as they completed their tasks. Most subjects eventually visited nearly all of the transitions in the model. However, the order and frequency with which certain paths were visited can be broken down into three styles of usage depending on the way the subject used the trainer:

1. Trainer as a backup
2. Trainer as a guide
3. Trainer as a validator

Subjects often switched between styles of use, particularly as the tasks became more difficult. All subjects switched to “Trainer as a guide” by the end of grade book setup task (Task 2.2 or 3.3).

5.1.1.1 Trainer as a backup In this style, the workflow is primarily driven by trial and error and/or prior knowledge. The user retains executive control throughout the majority of the session. If starting an unfamiliar task new task, the user may check trainer in the beginning to determine a starting point. After starting the task, the user starts most operations by searching through page, only referring to trainer upon getting stuck stuck. If the *what* component of the operation is discovered through the trainer but the *how* component is not, the user searches through the page in an attempt to discover the *how*. The overlay feature of the trainer is only invoked if searching does not work, which generally occurs only when the trainer is blocking the target element.

Figure 5.2 shows extracts of the model with the paths that represent two ways users discovered information about the Moodle interface colored. Users who used the trainer as a backup spent most of their time in the trial and error path in Figure 5.2(a), only occasionally switching to the trainer path in Figure 5.2(b) when they became stuck.

5.1.1.2 Trainer as a guide In this style, the user actively refers to the trainer through the process of completing a task. The trainer holds executive control throughout the majority of the task. The user generally begins each operation by referring to the trainer. They may use trial and error intermittently throughout the process, but this is not the driving method of discovery.

Users who used the trainer as a backup spent most of their time in the trainer path in Figure 5.2(b). They occasionally switched to the trial and error path in Figure 5.2(a), but tended to return to the trainer path frequently.

5.1.1.3 Trainer as a validator In this style, users completed most or all of a given task without using the trainer. This style tended to be used in conjunction with the "trainer as backup" style. After users executed a sequential set of operations, they referred to the trainer to check their work against the crossed-off steps in the trainer. In some cases, this prompted users to realize mistakes that may have occurred in previous tasks and led them to recheck tasks they previously thought to be complete.

Figure 5.3 shows a diagram of this style of use. After completing most or all of a given task, users returned to the "Did I complete the task correctly?" state. Instead of proceeding to the next operation, they returned to a previous operation and referred to the trainer to check if it was correct. If the operation was determined to be completed incorrectly, it was adjusted using the information gained from the trainer and executed again.

5.2 Difficulty of tasks and the trainer

Questions 1.3, 2.3, and 3.3 (Figures 4.4, 4.5, and 4.6) asked subjects to compare using the trainer between the calendar and assignment tasks (Question 1.3) and between the assignment and grade book tasks (Questions 2.3 and 3.3). For question 1.3, subjects responded either that they did not notice major differences between the use of the trainer for the two tasks or that they found the trainer to be more important to the assignment task, classifying the assignment task as more difficult. For questions 2.3 and 3.3, subjects unanimously found the trainer to be more important to complete grade book task, classifying the grade book setup as more difficult than creating assignments.

Subjects tended to agree that more "intuitive" tasks such as creating assignments could eventually be figured out without any aid by exploring the interface. However, many doubted their ability to complete the harder grade book task if they had been asked to complete it without having access to the trainer or other instructional material.

5.3 Best interface features

Table 5.1 shows a list of useful interface features highlighted as useful by subjects during the interview questions (Figures 4.4, 4.5, and 4.6). By this measure, the overlay and automatic crossing off of the steps were the most appreciated interface features. Subjects cited the overlay as particularly useful in situations where the trainer was blocking the element on the screen necessary to execute the operation. The crossing off of the steps was cited as helpful for rechecking previous operations and tasks as well as keeping track of where subjects were in the process of the current task.

Feature	Number of subjects
Overlay	4
Steps cross off	4
Move/minimize trainer	2
Steps correspond to interface	2
Some steps are optional	1

Table 5.1: Features highlighted by subjects as particularly useful during interviews

5.4 Highlighted uses of the trainer

Table 5.2 shows a list of uses of the trainer highlighted as useful by subjects during the interview questions (Figures 4.4, 4.5, and 4.6). When asked if they would incorporate the trainer into their workflow (Questions 1.2, 2.2, and 3.2), some users responded they would as they were learning to use Moodle, anticipating their use of the trainer to fade over time, like the scaffolding proposed by Wood et al. (1976).

Additionally, subjects identified several advantages of using the trainer over a traditional user guide or manual:

- Because the trainer is aware of the context, it offers an immediate answer to the question at hand. It can also quickly get users "unstuck" or give them reminders of information they may already know.
- It is faster to scan through the trainer than to read through documentation to solve a problem.
- It narrows down the interface items to just those essential to completing the task.

Use	Number of subjects
Fading scaffolding	4
Answers specific questions	3
Corrects errors	2
Gets "unstuck"	2
Faster than reading the manual	1
Narrows down interface options	1
Reminds of previous knowledge	1

Table 5.2: Uses of the trainer highlighted by subjects during interviews

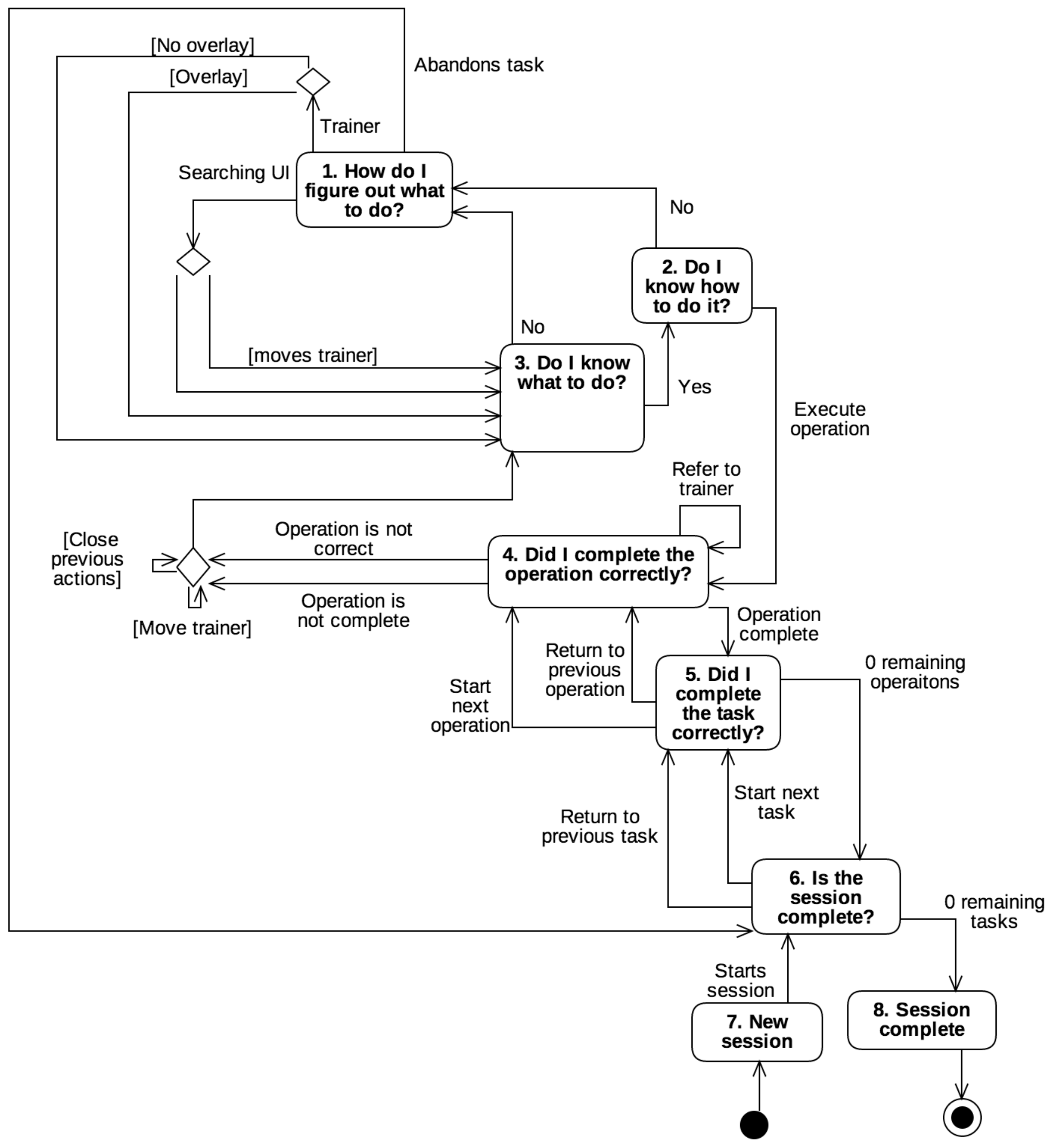
5.5 Requested improvements

Questions 1.4 and 3.4 (Figures 4.4 and 4.6) asked subjects about improvements they would like to see in the trainer. Table 5.3 lists the improvements users requested during the interviews. The most prevalent issue was the amount of space taken up by the trainer on the screen. Users found that the trainer blocked part of the interface and often needed to move or minimize it to complete portions of the tasks.

Another common issue was that some actions on the trainer were marked as complete before the subject began them. Some actions (such as creating graded and ungraded assignments) were similar enough that the trainer could not distinguish which action was being completed as it was keeping track in the background. Because of this, completing one of the actions would result in the steps of the other being marked as “complete” as well. This bug will be addressed in greater depth in the discussion.

Improvement	Number of subjects
Reduce the amount of space the trainer uses	3
Prevent the steps from crossing off before completion	2
Save trainer scrolling between pages	1
Clarify “category table” in “assign weights” action	1

Table 5.3: Improvements to the inline trainer requested by users during interviews



Step i. User begins in state 1, “How do I figure out what to do?”. Says, “so that’s going to be adding an entry, I guess?” Opens “add entry to calendar” action. “Yes.” Reads “scroll down to...” step. Transitions via “Trainer [No overlay].”

Step ii. User is in state 3, “Do they know what to do?” Transitions via “Yes.”

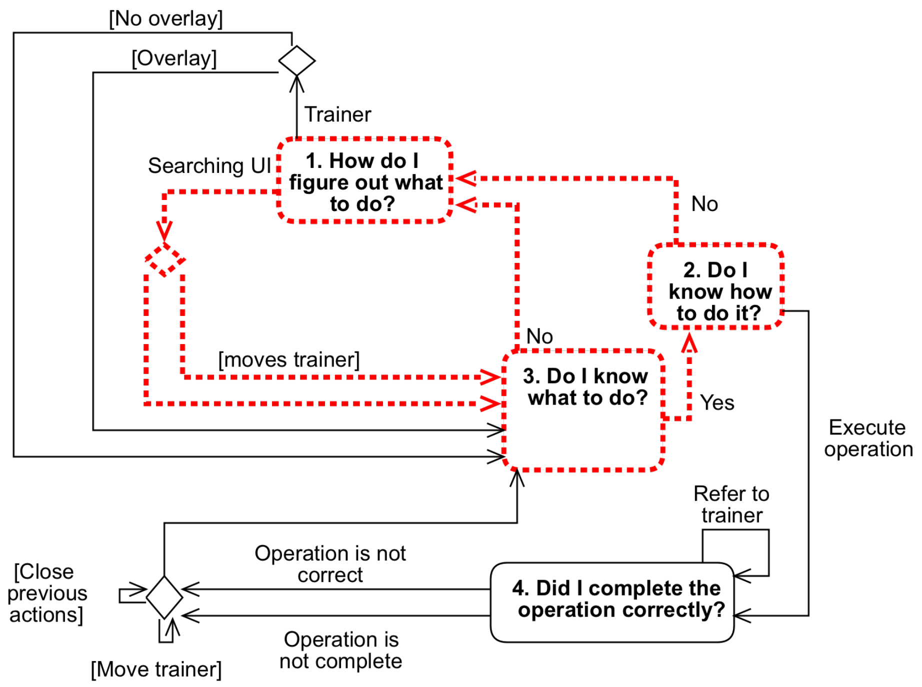
Step iii. User is in state 2, “Do I know how to do it?” Transitions via “No”.

Step iv. User is in state 1, “How do I figure out what to do?” Transitions via “Searching UI.” Scrolls down and searches for calendar. Finds calendar block.

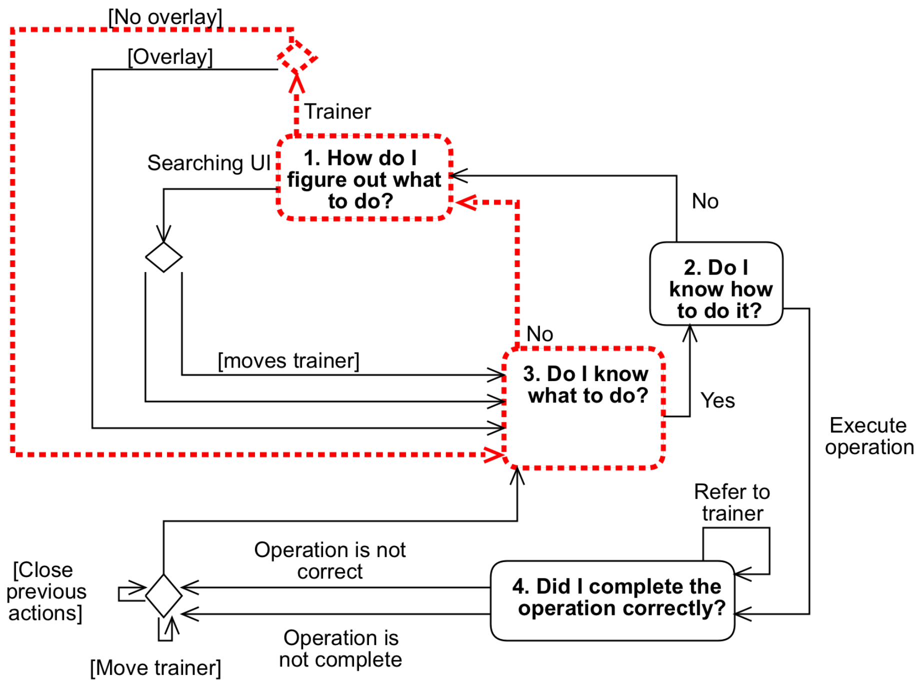
Step v. User is in state 3, “Do I know what to do?” Transitions via “Yes”.

Step vi. User is in state 2, “Do I know how to do it?” Transitions via “Execute operation.” Clicks the the title of the current month in calendar.

Figure 5.1: This model represents the process through which users complete the various tasks. Includes sample iteration through the model based on the transcript of Subject kbjk completing Task 1.1 (adding an entry to the course calendar).



(a) Discovery through trial and error



(b) Discovery through the trainer

Figure 5.2: Discovery process. These two excerpts from the trainer model (Figure 5.1) show the paths that represent two ways users discovered information about the Moodle interface. The paths for each discovery process are shown in red dashed lines.

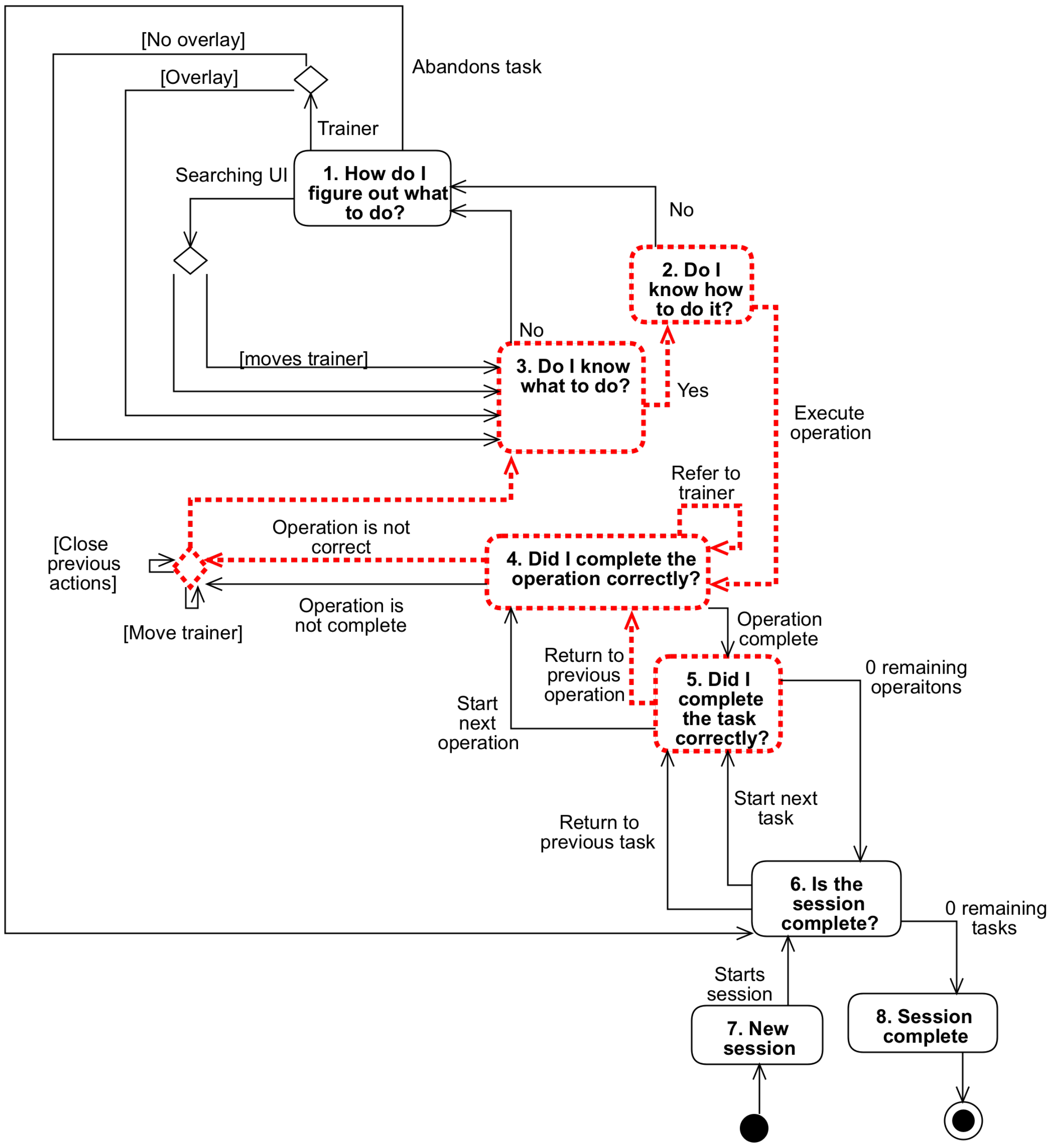


Figure 5.3: This model represents the path users took when using the trainer as a validator. The path that corresponds to this use style is shown in red dashed lines.

6 Discussion

6.1 Recommendations

Based on the observations in the pilot study, the inline trainer is a promising tool to help users learn to navigate and use the Moodle interface. This section will talk about specific recommendations for the use of the trainer beyond the pilot study.

6.1.1 Fine-tune checks for step completion

An essential function of the trainer is its ability to track the completion of steps at any given time, even when it is not open. Because of this, the trainer must "guess" which action is being worked on at any given time. For actions with similar steps, the trainer has difficulty making this distinction. For example, creating a graded and ungraded assignment differ by only a few steps. Because of this, both actions were often marked as complete when only one was actually intended by the user.

Additionally, several steps in the "assign weights [to the grade book]" action were marked as complete even if the user navigated to the wrong page. The page to set the grade aggregation mode for the *course* (the page on which the user needed to be in order to complete the step) is nearly identical to the page to set the grade aggregation for a grade *category.* When users navigated to the category page by mistake, the trainer led them to believe they were in the correct place and walked them through the next steps. Upon completing the steps on the category page and clicking the save button, users were redirected to the grade book page. Because the steps had not been completed in the right place, the trainer listed several previously-completed steps as incomplete which caused confusion for the user.

Both of these cases can be remedied by creating steps that are more fine-tuned to the task at hand. By looking for more specific contextual clues on the page and following more precise rules for tracking user activity in the background, the trainer can better determine which steps and actions have been completed by users.

6.1.2 Trainer in separate window

One common obstacle users faced was the amount of space taken up by the trainer on the screen. Because the trainer is an element of the same web browser window as the Moodle interface, it cannot be moved to a position in which it does not block some elements on the screen. The only way to show the entire Moodle interface is to collapse the trainer. After collapsing and reopening the trainer, subjects needed to reorient themselves and figure out where they left off before closing the trainer.

To combat this, the trainer should be modified in such a way that it can be displayed in a second browser window. This would allow the user to rearrange browser windows and place the trainer adjacent to the Moodle interface in a way that does not block any elements. The trainer would start as part of the same window as the Moodle interface as it does in the current design, with the added option to "pop" it out into a separate window to allow users to decide how they would to configure their workspace.

6.1.3 Find way to communicate conceptual model of Moodle

Some actions, such as the creation of and assignment of weights to grade book categories, required an understanding the internal structures and organization of Moodle. In order to assign weights to grade book categories, users must first change the grade aggregation method at the course level. This caused confusion, because the grade aggregation method can also be changed at the category level. The "edit" buttons for the course and category level were very similar and often led to confusion. Users assumed they should edit the category because they were trying to set a weight at the category level. Many users reported that setting the aggregation method at the course level had never occurred to them as an option, despite the text of the trainer step specifying to do so.

To combat this, the trainer should include a way to inform users of details like this in Moodle's internal structures and organization. This must be presented concisely so users understand the necessary information and are not overwhelmed or tempted to skip over it.

6.1.4 Minor User Interface (UI) improvements

Additionally, the researcher recommends the following minor improvements to the trainer user interface:

- The overlay feature should be expanded to allow multiple elements to be highlighted at once, as some steps can be completed via more than one element.
- The scroll top of the trainer should be saved across page loads. The current version of the trainer scrolls the action list up to the top at each page load, which means users must scroll back to the correct place each time.
- The trainer should include a section that allows users to search for actions by title and/or tags. This would allow quicker access than searching through the categories and subcategories, especially as more actions are introduced.

6.2 Further research

6.2.1 Use of overlay

The overlay was anticipated to be one of the most-used features during the design phase. Although users reported that they found the overlay feature to be useful when they needed it, they reserved it only for cases in which they felt completely stuck. Most preferred to search the interface before invoking the overlay. Further research could explore users' preferences for incorporating the overlay into their workflow and potential designs to make the overlay (or similar functionality) more relevant and useful. When do users invoke the overlay? What motivates them to do so (or not do so)?

6.2.2 Trainer and task performance

Further research could focus on the ways in which the trainer impacts users' abilities to complete tasks. How well do users complete tasks when they do or do not have access to the trainer? How does the trainer impact the completion of tasks that are performed consistently throughout the semester? How does it impact the completion of tasks that are primarily performed at certain points in the semester, such as the beginning or end?

7 Concluding Remarks

The inline trainer shows promise as a tool for instructors to learn to use Moodle as part of their normal workflow. Because of the constraints of the profession, teachers need to learn to use technology in a way that does not take their time away from performing other essential tasks. The changes and further research questions suggested can be used to turn the trainer into a robust tool that can be incorporated into live instances of Moodle.

7.1 Reflection

The overarching goal of this project was to design, implement, and test an inline trainer to help instructors learn to use Moodle. The path to achieving this goal resulted in a capstone for academic experience at Brandeis. It gave me an opportunity to apply many concepts and skills I learned in a theoretical capacity in a practical way. Here, I will reflect on the ways this project has incorporated each component of my undergraduate career.

7.1.1 Computer science

While developing the trainer plugin, I learned a lot about designing code for use by others. Most of my projects up until now have been solo projects where I am the only one who will maintain the code. The trainer was designed with the intention of being open source and eventually passed off to others to maintain. Because of that, I needed to plan out the code in a way that it would remain organized and clear for others in the future. I put a lot of thought into abstracting the code in such a way that extending the trainer can be done without modifying the core underlying mechanisms.

7.1.2 Teacher training for technology use

Conducting this study gave me insight into the ways people learn to use technology. I had previously completed coursework and reviewed literature about the theoretical frameworks surrounding this process, but this is the first time I consciously observed it in practice. I saw the ways in which people explored both the trainer and Moodle interface as a whole. I got a chance to observe people with different levels of comfort using technology and see the ways in which this affected their approach to figuring out how to complete the tasks. Watching people repeatedly try to perform the same tasks gave me insight into the the common ways people struggle and recover as well as the individual approach each person takes.

7.1.3 Cognitive design and engineering

One of the most surprising takeaways I had from this project was the amount I learned about the trainer from watching people use it. As the designer, I made assumptions about the ways people would use the tool. I assumed people would rely heavily on the overlay tool, when in fact they preferred to save it as a last resort. I had not considered the "trainer as a validator" style as an approach to completing the tasks. I saw the tool as a reference to be used during the execution of each task and did not consider its utility after the completion of a task. I learned that although I designed and built the tool, the only way to sufficiently understand its functionality is to observe others using it. I can use this insight to continue to improve and modify the trainer. As I design software in the future, I will be more aware of the importance of observing and soliciting feedback from users to better understand and improve its design.

7.1.4 Research

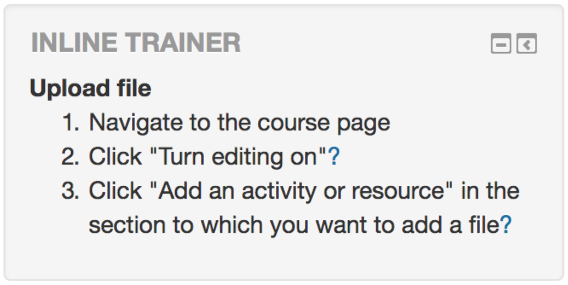
Before undertaking this project, I understood "research" as a vague, nebulous concept and was unclear on the actual process. I had seen the results of prior research in the form of published papers, but did not know what needed to be done to get to that point. As I learned at each step, from designing the experiment to analyzing and writing up my findings, the process became more and more enjoyable. One of my biggest struggles with the research component was that there was not a pre-determined answer toward which I was working. There are an infinite number of ways to analyze and interpret the data, and given unlimited time I do not know if I would ever reach a point of feeling as though I had covered every possible angle. Completing this project has given me a new appreciation for and fascination with research. I have a new set of tools with which to measure the impact of my future endeavors in the world of educational technology.

Appendices

A Technical design of the inline trainer plugin

A.1 Proof of concept

At the beginning of this project, a Moodle block plugin was created to test whether a trainer that sits on the edge of the screen and observes events on the page would be technically viable (Figure A.1). The plugin is very simple. It includes only three steps towards an ”Upload file” action. It is not movable on the screen and can only be collapsed via the block editor menu. The plugin included early version of the main features such as the crossing off of completed steps and the overlay functionality.


INLINE TRAINER

Upload file

1. Navigate to the course page
2. Click "Turn editing on"?
3. Click "Add an activity or resource" in the section to which you want to add a file?

Figure A.1: A simple Moodle block plugin created to test whether an inline trainer was technical viable

---

A concern with the development of the plugin was finding a way to hook into Moodle’s JavaScript to observe the events on the page (such as clicks, key presses, etc.). Although Moodle provides a JavaScript API, it was decided that this API would limit the abilities of the trainer. The API was used to load the plugin once the Moodle page was loaded, but the logic within the trainer was separate. The trainer primarily uses the jQuery[5] library because of its ability to easily access all elements of an HTML document. An advantage to this approach is that the trainer is relatively self-contained and does not rely on other developers implementing the Moodle JavaScript API. The plugin does not need to “know” the way any part of the Moodle User Interface (UI) is set up from a programming standpoint, it just needs to check the resulting elements on the page. A disadvantage is that because the trainer is unable to access Moodle’s JavaScript, it cannot subscribe to dynamic events (e.g. opening a modal window). To combat this, certain trainer steps include short delays (10ms) before running certain functions to ensure the Moodle interface has had time to finish rendering after responding to an event. While this approach seems to be successful in most cases, it is a glorified “guess-and-check” system that is susceptible to error if any of the Moodle JavaScript experiences a major lag. This jQuery-based approach to monitoring the page was adopted into the final version of the trainer plugin.

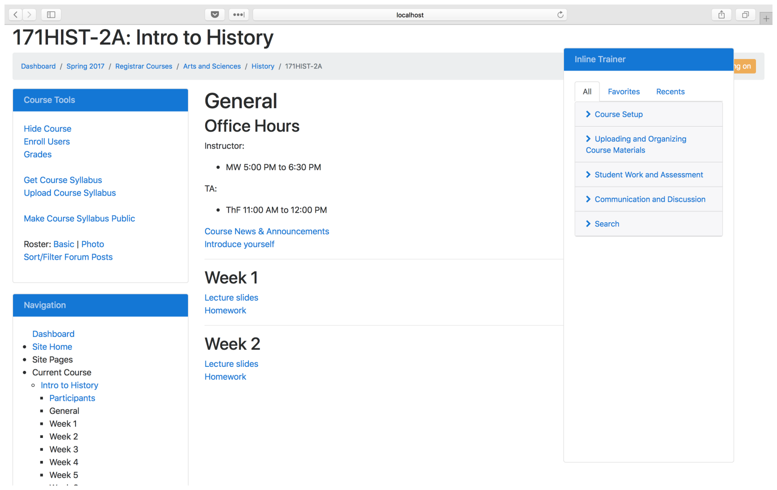
The original proof of concept plugin is available on Github[6]

A.2 Trainer prototype

After the proof-of-concept, a prototype of the trainer was created to quickly test the usability of the interface (Figure A.2). The prototype was created using TypeScript compiled into JavaScript.

[5]https://jquery.com/
[6]https://github.com/zakkolar/moodle-block_inlinetrainer-prototype

The layout and UI used the Angular[7] framework. The prototype included a mockup of a simplified version of Moodle to allow users to test the trainer in the context of a large application.



localhost

171HIST-2A: Intro to History

Dashboard / Spring 2017 / Registrar Courses / Arts and Sciences / History / 171HIST-2A

Course Tools

Hide Course
Enroll Users
Grades

Get Course Syllabus
Upload Course Syllabus

Make Course Syllabus Public

Roster: Basic | Photo
Sort/Filter Forum Posts

Navigation

Dashboard
- Site Home
- Site Pages
- Current Course
  - Intro to History
    - Participants
    - General
    - Week 1
    - Week 2
    - Week 3
    - Week 4
    - Week 5

General

Office Hours

Instructor:

- MW 5:00 PM to 6:30 PM

TA:

- ThF 11:00 AM to 12:00 PM

Course News & Announcements
Introduce yourself

Week 1

Lecture slides
Homework

Week 2

Lecture slides
Homework

Inline Trainer

All | Favorites | Recents

- Course Setup
- Uploading and Organizing Course Materials
- Student Work and Assessment
- Communication and Discussion
- Search

Figure A.2: The prototype used to test the design of the trainer

---

The prototype expanded on the structure of the proof-of-concept and introduced the relationship between actions and steps. It built upon the trainer's ability to observe the page and check for step completion, which is detailed later in this appendix. Much of the underlying code for the steps and actions from the prototype was incorporated into the final plugin.

The code for the prototype of the trainer is available on Github[8]

A.3 Trainer components

A.3.1 Moodle Plugin Interface

The part of the plugin written to interface with the backend of Moodle is written in PHP. This constitutes a small portion of the plugin, including the following features:

- Installing/uninstalling the plugin
- Creating/updating the database structure
- Customizing plugin settings via Moodle admin interface
- Storing/retrieving user information (e.g. favorites, open/closed status of the trainer, consent to participate in study, etc.)
- Storing/retrieving research data collection
- Attaching the plugin to each page

A.3.2 User Interface

The majority of the plugin was written in TypeScript which was compiled to JavaScript using Webpack. The trainer interface was created using the Vue[9] framework for JavaScript. Vue was selected because of its two-way binding and relatively simplicity compared to Angular. The two-way binding means the framework links the data to the interface, so the primary focus of development could remain on the logic of the trainer itself.

[7] https://angular.io/
[8] https://github.com/zakkolar/inlinetrainer-prototype
[9] https://vuejs.org/

A.4 Programatic design

The classes most central to the functionality of the trainer were the Action and Step classes, shown in Figure A.3. The remainder of this section will focus primarily on the design of the mechanism through which the trainer keeps track of the steps and their completion. See section A.5 for a link to the Github repository in which the plugin is hosted for more information about the code.

A.4.1 Requirements for the steps

As the technical design was planned for the inline trainer steps, several goals and challenges arose:

- The trainer should determine when steps are completed/uncompleted by observing the page as the user interacts with it
- The trainer should infer which steps have already been completed by the user based on the state of the page
- For steps whose completion cannot be inferred, the trainer must remember if they have been completed

These requirements shaped the technical design of the trainer. The functional nature of JavaScript greatly helped the implementation of these requirements. Steps stored functions defined at their instantiation as properties, which provided flexibility for each step to interact with the page in its own way.

A.4.2 Step completion

One of the most essential features of the trainer is its ability to track the completion (or uncompletion) of steps. In the context of the trainer, to "uncomplete" a step is to undo the completion of that particular step (e.g. if a step is completed when the user opens a popup window, "uncompleting" that step would be closing the window). "Incomplete" is used to describe a step that has not been completed. The most basic level of the detection of step completion takes place within five function properties of the `Step` class:

`checkCompleteFunction` This function checks to determine if the step has been completed. Depending on the step, this may check for the existence of an element on the page, the value of an input, the URL of the current page, or any number of other properties.

`watchCompleteFunction` This function attaches a listener to the page to detect when the step may be complete and invokes the `checkCompleteFunction` to make the determination. This is useful for steps that can be completed after the page is loaded, such as filling in a form field.

`watchUncompleteFunction` This function is similar to `checkCompleteFunction`, but serves the opposite purpose. After a step is complete, the `watchUncompleteFunction` function attaches a listener to detect when the completion may have been uncompleted. Upon being triggered, the listener executes the `checkCompleteFunction` to confirm.

`unwatchCompleteFunction` This function removes the watcher set by `watchCompleteFunction` when an action is determined to be complete. Because the completion of an action may trigger other events within the trainer (such as updating the current step, setting watchers for the next step, etc.), it should only be reported once. Removing the watcher immediately after an step is completed prevents the same step from being marked as complete multiple times in a row.

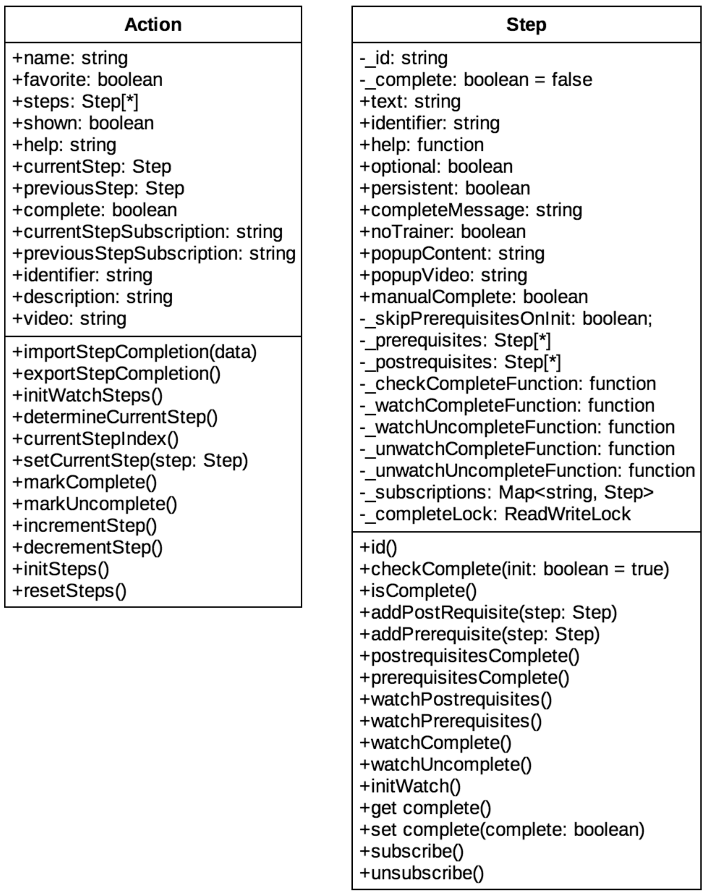


Figure A.3: Class diagram for the Action and Step classes, two classes most important to the functionality of the trainer

---

`unwatchUncompleteFunction` Similar to `unwatchCompleteFunction`, this function removes the watcher set by `watchUncompleteFunction` when an action is determined to have been uncompleted.

Figure A.4 shows a diagram of the relationship between these functions.

A.4.3 Prerequisites and Postrequisites

Because the trainer is added to Moodle via JavaScript, it is reinitialized on every page load. This means that it needs to infer the completion of steps completed on previous page loads that it may not have observed directly. In some cases, the `checkCompleteFunction` is sufficient to accomplish this. Even if the trainer cannot watch the completion of the step, the page is altered in such a way that `checkCompleteFunction` is able to determine that the step has been completed. For example, clicking the "turn editing on" adds several indicators to the page that editing mode is enabled. Even if the page is reloaded, these indicators are present until editing mode is turned off.

In other cases, `checkCompleteFunction` is not sufficient. For some steps, evidence of completion is only available on certain pages. If the page that shows such evidence is not visible, the `checkCompleteFunction` cannot determine if the step has been completed. Take creating a graded

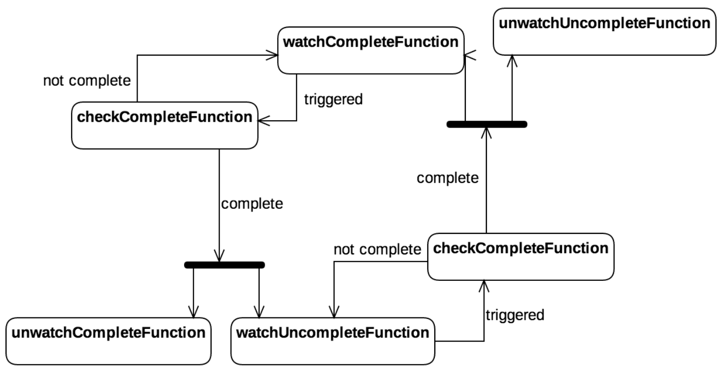


Figure A.4: The cycle of completion functions for a step in the trainer

assignment as an example. The beginning steps are summarized as follows:

1. Navigate to the course page
2. Click "Turn editing on"
3. Click "Add an activity or resource"
4. Select "assignment" as the activity type
5. Click "add"
6. Type your assignment's name

The first four steps are completed on the course page. While the user is on the course page, the trainer can detect whether each of these steps has been completed by observing the page. Even if a step is not completed in the same page load, for example, if the user previously enabled the editing mode, the trainer can observe the state of the page to determine that the step has been completed at some point. But as soon as the user clicks the "add" button in step five, they are redirected to a new page to fill in the details of the assignment. Once this assignment page is loaded, the trainer is unable to tell whether the steps that take place on the course page have been completed.

Although the trainer is unable to directly observe the completion of some steps, it can make inferences if it "understands" the relationship between steps. Postrequisites and prerequisites were introduced to help the trainer with this understanding.

Let Step A be some step for an action in the trainer. Steps who have Step A as a postrequisite are implied to be complete if and only if Step A is also complete. In other words, if we observe Step A to be complete, we can assume any step with Step A as a postrequisite to be complete as well. If a step has multiple postrequisites, just one must be complete to infer the completeness of the original step.

In the above example, the completion of step 5 is determined by comparing the URL of the current page to the URL of the assignment details page. If they match, step 5 is complete. Steps 1 - 4 all have step 5 as a postrequisite. This means that although the trainer cannot see evidence of their completion on the assignment page, it can safely assume that they have all been completed.

Prerequisites are similar to postrequisites, but are used to determine incompleteness. Step A is implied to be incomplete if and only if at least one of its prerequisites is incomplete.

A.4.4 Determining the completeness of a step

Because of the prerequisite and postrequisite properties, just executing the `checkCompleteFunction` is not enough to determine the completeness of a step. The algorithm used to determine if a step is complete is presented here in pseudocode:

```
# check if we can directly observe the step as complete
observedComplete <- checkCompleteFunction() # boolean

# check if the prerequisites are all complete
prerequisitesComplete <- true
for each prerequisite do
        if prerequisite is not complete do
                prerequisitesComplete <- false

# check if at least one postrequisite is complete
postrequisitesComplete <- false
for each postrequisite do
        if postrequisite is complete
                postrequisitesComplete <- true

if prerequisitesComplete = false
        step is not complete
else if observedComplete = true or postrequisitesComplete = true
        step is complete
```

In the trainer, this algorithm is implemented with promises and callbacks to be used asynchronously. For the sake of clarity, it is presented synchronously here.

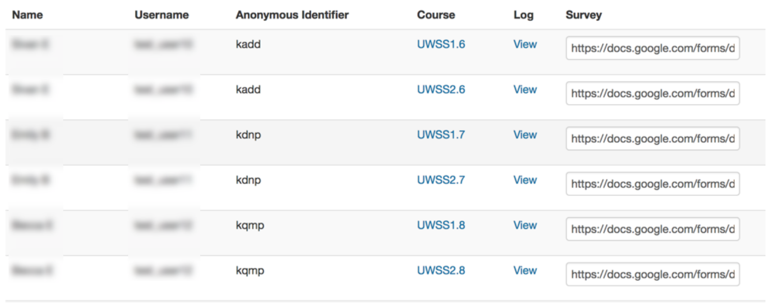
A.5 Plugin repository

The code for the plugin has many more features and considerations than there is space to include or even describe here. However, it has been made open-source and is available on Github[10]. The branch FINAL_RESEARCH_CODE will not be updated to preserve the state of the code during the research.

[10] https://github.com/zakkolar/moodle-local_inlinetrainer/tree/FINAL_RESEARCH_CODE

B Moodle plugin for experiment administration

The administration of experiments required many repeated tasks, including creating user accounts, creating courses, populating the courses with sections, assigning users to roles in the courses, and giving users the correct capabilities to use the inline trainer. A Moodle plugin was created to help automate this process to save time and reduce the likelihood of error in the setup for the trials (Figure B.1). The plugin provided a list of users and their corresponding courses. It generated a link for them to fill out the demographic survey that automatically included their unique identifier. It showed their real name (only for the purposes of keeping track during the experiment phase) as well as an anonymous identifier to distinguish them within the dataset without revealing their identity. The plugin's settings allowed the proctor to set default information such as the number of courses to be created for each user, the course naming scheme, the username scheme, the default sections to be shown in the course, and the roles that should be given to users upon creation.



Name	Username	Anonymous Identifier	Course	Log	Survey
[redacted]	[redacted]	kadd	UWSS1.6	View	https://docs.google.com/forms/d
[redacted]	[redacted]	kadd	UWSS2.6	View	https://docs.google.com/forms/d
[redacted]	[redacted]	kdnp	UWSS1.7	View	https://docs.google.com/forms/d
[redacted]	[redacted]	kdnp	UWSS2.7	View	https://docs.google.com/forms/d
[redacted]	[redacted]	kqmp	UWSS1.8	View	https://docs.google.com/forms/d
[redacted]	[redacted]	kqmp	UWSS2.8	View	https://docs.google.com/forms/d

Figure B.1: A table of experiments displayed by the experiment administration plugin

The source code for the plugin can be found on Github[11].

[11] https://github.com/zakkolar/moodle-local_inlinetrainer_experiments

C Web-based tool analyze Atlas.TI project

Atlas.TI is a powerful tool for tagging qualitative data, but does not offer an extensive set of analysis tools. However, projects can be exported in XML format. Because of this, a web-based tool was created to parse the XML, convert it to JavaScript, and manipulate and display the data in useful ways. The tool was created using JavaScript and the Vue framework.

The source code for this tool can be found on Github[12].

[12]https://github.com/zakkolar/atlasTI-analysis

D Changes between the first and second rounds

After observing and gathering feedback from subjects, the following changes were made to the Inline Trainer interface to attempt to reduce errors and confusion:

D.1 "Start Over" button

In some cases, the trainer was unable to distinguish which action the user was attempting to complete and marked multiple sets of steps as complete simultaneously. This led to confusion when subjects saw that certain steps were already marked as complete when they had not yet begun the second of the two similar tasks.

The "Start over" button was originally placed below each action with the intention for users to see it as they scrolled down and completed an action. However, when subjects came across a new action that was already marked as complete, they generally did not scroll down low enough to see the "Start over" button. To make the button more visible in this situation, the "Start over" button was displayed a second time above the action when all of its steps were marked as complete.

The text of the button was changed from "Start over" to "(click here to start over)" in an attempt to better distinguish the text as a clickable link.

D.2 Assigning weights to categories

The actions inside of the "Grade book setup" caused confusion to subjects. One source of confusion was the separation of the creating and assigning weights to grade book category actions. The task list presented a list of categories with their corresponding weights as they would appear on a syllabus. Moodle automatically adjusts the weights to total at 100. Because of this, users must create all grade categories before assigning weights so that their sum can be adjusted to 100.

Most users attempted to create the categories and assign category weights at the same time. They often began in the "Create grade categories" action in the trainer (the optimal starting place to complete this task) and then opened the "Assign weights" category and attempted to complete the steps for the second action before finishing the first. To prevent this, the "Assign weights" action was given a new first step that instructed the users to "Create all of your grade categories before assigning weights." Because there is no way for the trainer to detect when the user has created all necessary categories, the step included a link for users to click upon completion to signal the trainer to move onto the next step in the action.

Additionally, users were often confused by the "At the top of the category table, find the name of your course. On the right of that row, click 'Edit' and then 'Edit settings"' step in the "Assign weights" step. Rather than clicking the "Edit settings" link at the very top of the course, users would often click "Edit settings" next to one of the categories that had just been created. The text, "to edit the course-level settings" was added to the end of the existing description in an attempt to clarify this.

E Experiment materials

E.1 Pre-experiment survey questions

The following online survey was given to subjects before their participation in the experiment.

What is your primary role in classes?
a. Instructor
b. Teaching assistant
c. Student
d. My position is outside of the classroom
e. Other (please specify)

What is your role at Brandeis University?
a. Professor/faculty
b. Department Staff
c. Undergraduate student
d. Master's Student
e. PhD Student
f. Other (please specify)

Which of these statements do you most identify with?
a. I am excellent with technology; I can help others use it
b. I'm very comfortable with technology
c. I can use technology
d. I can use basic technology, but in a limited way
e. I'm uncomfortable with technology

How many semesters have you used LATTE at Brandeis?
a. This is my first time
b. 2-3 semesters
c. More than 3 semesters

Which of the following actions have you completed using LATTE? Mark all that apply.
a. Uploading course syllabus
b. Managing course readings
c. Creating assignments
d. Evaluating/scoring assignments
e. Submitting assignments
f. Reviewing feedback on assignments
g. Posting documents
h. Setting up the grade book
i. Recording grades in the gradebook
j. Viewing grades in the gradebook

E.2 Task handout for Session 1.1

Tasks (day 1)
Suppose you are an instructor for a writing course. Your task will be to set up the gradebook and create a few assignments to start the semester.

1. To start this task, we need to add our first lecture to the course calendar. Create an entry in the course calendar with the following details:
   Event title: UWS Lecture
   Date: 21 March 2018

2. Oops! We forgot to add the duration. Edit your calendar entry and set the duration to 120 minutes so students know what they've gotten themselves into.

3. Next, we need to create some graded assignments so students can submit their work for grades:
   a. Create a graded assignment under the "Close Read" section of the main course page with the following details:
      Assignment name: Close Read Draft
      Description: Submit a draft of your close read essay.
      Due date: 15 April 2018
      Maximum grade: 60 points
   b. Create another graded assignment under the "Close Read" section of the main course page with the following details:
      Assignment name: Close Read Draft
      Description: Submit a revised draft of your close read essay
      Due date: 18 April 2018
      Maximum grade: 60 points

4. Finally, we need to create a pre-draft assignment for students to submit. This assignment will not be graded; but students will receive written feedback from you after submission.
   a. Create an ungraded assignment under the "Lens Analysis" section of the main course page with the following details:
      Assignment name: Lens Pre-Draft 1
      Description: Write a paragraph summarizing your lens text.
      Due date: 21 April 2018
   b. Create another ungraded assignment under the "Lens Analysis" section of the main course page with the following details:
      Assignment name: Lens Pre-Draft 2
      Description: Write a paragraph summarizing your analysis text.
      Due date: 25 April 2018

E.3 Task handout for Session 1.2

Tasks (day 2)

Imagine you are a University Writing Seminar instructor. Your task will be to set up the gradebook and create a few assignments to start the semester.

1. First, we need to create some graded assignments so that students can submit their work for grades.
    a. Create a graded assignment under the "Close Read" section of the main course page with the following details:
        Assignment name: Close Read Essay
        Description: Submit your final Close Read Essay.
        Due date: 19 April 2018
        Maximum grade: 90 points
    b. Create another graded assignment under the "Close Read" section of the main course page with the following details:
        Assignment name: Close Read Revision
        Description: Submit any revisions you wish to make based on my feedback.
        Due date: 27 April 2018
        Maximum grade: 90 points

2. Next, we need to get the gradebook ready for the semester. Configure the gradebook to accept the following categories and weights for the given syllabus:
    Pre-draft work: 20%
    Rough drafts: 30%
    Final drafts: 50%

E.4 Task handout for Session 2.1

Tasks

Suppose you are an instructor for a writing course. Your task will be to set up the gradebook and create a few assignments to start the semester.

1. To start this task, we need to add our first lecture to the course calendar. Create an entry in the course calendar with the following details:
   Event title: UWS Lecture
   Date: 21 March 2018
2. Next, we need to create some graded assignments so students can submit their work for grades:
   a. Create a graded assignment under the “Close Read” section of the main course page with the following details:
      Assignment name: Close Read Draft
      Description: Submit a draft of your close read essay.
      Due date: 15 April 2018
      Maximum grade: 60 points
   b. Create another graded assignment under the “Close Read” section of the main course page with the following details:
      Assignment name: Revised Close Read Draft
      Description: Submit a revised draft of your close read essay
      Due date: 18 April 2018
      Maximum grade: 60 points

3. Next, we need to get the gradebook ready for the semester. For this task, you can ignore the assignments you created previously. Configure the gradebook to accept the following categories and weights for the given syllabus:
   Pre-draft work: 20%
   Rough drafts: 30%
   Final drafts: 50%

References

Bødker, S. (1995). Context and consciousness. chapter Applying Activity Theory to Video Analysis: How to Make Sense of Video Data in Human-computer Interaction, pages 147–174. Massachusetts Institute of Technology, Cambridge, MA, USA.

Bostrom, R. P., Olfman, L., and Sein, M. (1986). Training for end-user computing: Are basic abilities enough for learning? In *Proceedings of the Twenty-second Annual Computer Personnel Research Conference on Computer Personnel Research Conference*, SIGCPR '86, pages 1–11, New York, NY, USA. ACM.

British Educational Communications and Technology Agency (Becta) (2004). A review of the research literature on barriers to the uptake of ict by teachers: British educational communications and technology agency (becta).

Burns, M. and Dimock, V. (2007). *Technology as a catalyst for school communities : beyond boxes and bandwidth.* Rowman & Littlefield Education, Lanham.

Costello, E. (2013). Opening up to open source: looking at how moodle was adopted in higher education. *Open Learning*, 28(3):187 – 200.

Dexter, S. L., Anderson, R. E., and Ronnkvist, A. M. (2002). Quality technology support: What is it? who has it? and what difference does it make? *Journal of Educational Computing Research*, 26(3):265–285.

Granger, C., Morbey, M., Lotherington, H., Owston, R., and Wideman, H. (2002). Factors contributing to teachers' successful implementation of it. *Journal of Computer Assisted Learning*, 18(4):480–488.

Inan, F. A. and Lowther, D. L. (2010). Factors affecting technology integration in k-12 classrooms: A path model. *Educational Technology Research and Development*, 58(2):137–154.

Jin, J. and Dabbish, L. A. (2009). Self-interruption on the computer: A typology of discretionary task interleaving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1799–1808, New York, NY, USA. ACM.

Kirkwood, M., Van Der Kuyl, T., Parton, N., and Grant, R. (2000). The new opportunities fund (nof) ict training for teachers programme: Designing a powerful online learning environment. In *European Conference on Educational Research.*

Krisler, B. (2014). *Continuous Software Training with Three Inline Trainers.* PhD thesis, Brandeis University.

Krisler, B. and Alterman, R. (2008). Training towards mastery: overcoming the active user paradox. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, pages 239–248. ACM.

Krisler, B. and Alterman, R. (2018). Inline training: a technique for continuous, within-task learning. *Research in Learning Technology*, 26.

Laoledchai, Y., Pek, L. W. L., and Low, G. (2008). Improving the effectiveness of end-user training outcomes. In *ACIS 2008 Proceedings - 19th Australasian Conference on Information Systems.*

McKenzie, J. (2001). *How teachers learn technology best.*

Norman, D. A. (1991). Cognitive facts. In Carroll, J. M., editor, *Designing interaction: Psychology at the human-computer interface*, volume 4. CUP Archive.

Peck, C., Mullen, C. A., Lashley, C., and Eldridge, J. A. (2011). School leadership and technology challenges: Lessons from a new american high school. In *AASA Journal of Scholarship & Practice*, volume 7. American Association of School Administrators.

Perkins, D. N. (1997). Person-plus: A distributed view of thinking and learning. In Salomon, G., editor, *Distributed cognitions: Psychological and educational considerations.* Cambridge University Press.

Thorsteinsson, G. and Niculescu, A. (2012). Examining teachers' mindset and responsibilities in using ict. *Studies in Informatics and Control*, 22(3):315–322.

Toprakci, E. (2006). Obstacles at integration of schools into information and communication technologies by taking into consideration the opinions of the teachers and principles of primary and secondary schools in turkey. 9:1–16.

VanLehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology*, 47(1):513–539. PMID: 15012487.

Wood, D., Bruner, J. S., and Ross, G. (1976). The role of tutoring in problem solving*. *Journal of Child Psychology and Psychiatry*, 17(2):89–100.

Zhao, Y. and Cziko, G. A. (2001). Teacher adoption of technology: A perceptual control theory perspective. *Journal of Technology and Teacher Education*, 9(1):5–30.