

FUNCTIONAL INTERRUPTION
MANAGEMENT SYSTEMS

FUNCTIONAL INTERRUPTION MANAGEMENT SYSTEMS

By CHINMAY SHETH,

A Thesis Submitted to the School of Graduate Studies in Partial
Fulfillment of the Requirements for
the Degree MSc, Computer Science

McMaster University © Copyright by Chinmay Sheth, February 2025

McMaster University

MSC, COMPUTER SCIENCE (2025)

Hamilton, Ontario, Canada (Dept of Computing and Software)

TITLE: Functional Interruption Management Systems

AUTHOR: Chinmay Sheth
BASc (Computer Science),
McMaster University

SUPERVISOR: Dr. Christopher K Anand

NUMBER OF PAGES: xiii, 149

Abstract

Coding mentors must gauge when a student is engaged or not in their primary task before interrupting them. Gauging student engagement is a difficult task when one mentor is responsible for mentoring many students in a classroom; at times, mentors have taught 30 students in one classroom. An inopportune interruption leads to disruption and lowers performance on the primary task a student is attempting to complete, whereas failing to provide timely support could result in a student giving up on the lesson. A programmatic approach to interrupting students would be less disruptive, especially in a virtual setting, and would enable mentors to focus on more “difficult” queries from students. Synthesizing the mentors’ experiences as well as a literature review we formulate our goal as: **How might we interrupt students for announcements or to collect feedback, such that it results in the least amount of disruption to the interrupted student?** The primary solution offered is the design and implementation of an Interruption Management System (IMS) in the functional programming language, Elm. An IMS helps users to effectively perform their primary task by minimizing the disruptive nature of an interruption. This research explores the non-trivial task of detecting when a user is engaged in their primary task, when to interrupt them such that there is minimum disruption, and how to determine how “disruptive” an interruption is to a user. Two iterations of an IMS prototype design and implementation, and an evaluation plan for the prototype

are presented to answer the aforementioned motivating problem.

Dedication

There are a lot of people and things that I am grateful for, which were necessary for the completion of this thesis.

My wife Dhyani has shown unwavering support to me throughout these past few years. Working full time and attempting to complete a thesis was only possible because Dhyani lifted and supported me throughout the process. Only she is aware of the details and has kept me sane this entire time. This thesis should be hers as much as it is mine. Thank you my love.

I was raised by immigrants who sacrificed their careers, friends, and family to give me a better life. Their love and support never allowed any material shortcomings to surface. The completion of this thesis is a trophy of their foresight and endurance. Thank you mom, dad, and my sister, Shama.

Finally, thank you to my supervisor, Dr Christopher Anand who took me into his lab in my second year of undergraduate studies, and whom I've stuck with ever since. Aside from being a talented researcher, Dr Anand is also the most humble person I've ever met. I become a better researcher by simply being in his presence and absorbing his thoughts and writing. Thank you for everything.

Table of Contents

Abstract	iii
Dedication	v
Abbreviations	xiii
1 Introduction	1
1.1 ACM SIGSOFT Empirical Standards	1
1.2 STaBL Foundation	4
1.3 Motivating Problem	5
1.4 Chapter Overview	7
1.5 Contributions	8
2 Literature Review	9
2.1 Classification of Interruptions	10
2.2 Disruptiveness of Interruptions	21
2.3 Interruption Management Systems	26
3 Methodology	31
3.1 Design Thinking Primer	31

3.2	Designing the First Iteration of the Prototype	37
3.3	Research Study Design	44
4	Software Technology Stack	55
4.1	Elm & Model-View-Update	56
4.2	GraphicSVG	56
4.3	TEASync	58
4.4	State-Diagram Draw	59
4.5	Teacher Dashboard & MultiShapeCreator	64
5	Prototype Iteration #1	77
5.1	Teacher Screen	78
5.2	Student Screen	81
5.3	Interruption Classifications	85
5.4	Requirements Delivery	88
5.5	STaBL.rocks Proposed Usage Example	92
5.6	Types	94
6	Results of Research Cycle #1	97
6.1	Feedback Collection Setup	97
6.2	Feedback Components	98
6.3	Prototype Alternatives	101
7	Prototype Iteration #2	105
7.1	Implementation	106

8 Results of Research Cycle #2	111
8.1 General Feedback	112
8.2 Themes	113
9 Prototype Iteration #3	116
9.1 Historical Announcement Log	116
9.2 Personified Emoji Aesthetic Enhancements	118
10 Future Work	122
10.1 Gap Analysis	122
10.2 Prototype Updates	124
11 Conclusion	128
A Question Classification Text	131
B Interruption Management System Prototype Types, States, and Mes-	
sages	136
B.1 Local Messages	136
B.2 Local State	137
B.3 Global Messages	137
B.4 Type Aliases	137

List of Figures

1.1	STaBL.rocks IDE Screenshot	5
2.1	Interruption Management State Model	17
2.2	IRC Framework	21
2.3	Seven Stages of Action	23
3.1	Five Stages of Design Thinking	34
3.2	Double Diamond Framework	36
4.1	Model-View-Update	57
4.2	Screenshot of STaBL.rocks IDE with Compiled Code	58
4.3	State-Diagram Draw Implementation Screenshot	62
4.4	State-Diagram Draw “Waiting” State and Button to Transition to “HelpInstructions” State	63
4.5	State-Diagram Draw non-Waiting or HelpInstructions State	64
4.6	Teacher Dashboard Landing Page	66
4.7	Teacher Dashboard Login Page	67
4.8	Teacher Dashoard New Logins Page	68
4.9	Teacher Dashboard Screenshot for New Logins	69
4.10	Teacher Dashboard Student Welcome Page	70
4.11	Four screenshots capturing the various stages of the Teacher Dashboard	71

4.12	Teacher Dashboard MultiShapeCreator Screenshot	72
4.13	MultiShapeCreator Landing Page Screenshot	73
4.14	MultiShapeCreator Default Page Screenshot	74
4.15	MultiShapeCreator Hierarchical Task Analysis	75
4.16	MultiShapeCreator Curve Example	76
5.1	Interruption Management System Landing Screenshot	79
5.2	Interruption Management System Teacher Screen	81
5.3	Interruption Management System Student Screen	82
5.4	Interruption Management System Teacher Screen Filled Out	83
5.5	Software Tool Personification	84
5.6	Student Screen Screenshot	85
5.7	Student Screen with Notification Screenshot	86
5.8	Announcement Test Screenshot	87
5.9	Bounded Deferral Timer Screenshots	88
5.10	Teacher UI State Diagram	95
5.11	Student UI State Diagram	96
7.1	Prototype 2: Thought Bubbles on Emoji	109
7.2	Prototype 2: Paw	110
9.1	Opening the Historical Announcements Log	117
9.2	Historical Announcements Log	119
9.3	Historical Announcements Log Expiry Timer Input	120
9.4	Prototype Iteration # 3 Emoji	121

List of Tables

1.1	Lists the Attributes that are required in the Design Science Standard and a reference to which section they are found in the thesis. A detailed list of the attributes and definitions for the Design Science standard are available in [43].	4
2.1	Categories of interruptions.	13
2.2	Examples of the Interruption (I), Reaction (R), Comprehension (C) framework with low (0) or high (1) values to capture blending of dimensions. Willermark et al. [50] have developed the polite pop-up, which is a “modal pop-up that is created based on click events.” This allows the user to choose to interact with the pop-up rather than automatically placing a pop-up on-top of the users’ display like a traditional pop-up.	22
2.3	Experiment types in the area of interruption management systems in a desktop environment.	29
3.1	Requirements table for developing the initial prototype for the STaBL Rocks Interruption Management System.	39

3.2	Attributes that are required in the Action Research Standards and a reference to which section they are found in the thesis. Some attributes are not applicable, marked using “N/A”, because an empirical study, which has not been performed in this thesis, is required to determine the necessary information for the attributes.	44
5.1	Captures the initial delivery requirements for the prototype and determines whether they have been met by the alternative solution. . . .	89
6.1	Captures the initial delivery requirements for the prototype and determines whether they have been met by the alternative solution. . . .	102

Abbreviations

IDE	Integrated-Development-Environment
HMW	How Might We
IM	Instant Messaging
HTA	Hierarchical Task Analysis
UI	User Interface
IMSM	Interruption Management State Model
IRC	Interruption-Reaction-Comprehension
HCI	Human-Computer Interaction
IMS	Interruption Management Systems
MD	Managing Director
MVU	Model-View-Update
GUI	Graphical User Interface
TEA	The Elm Architecture
DOM	Document Object Model
LG-MVU	Local-Global Model-View-Update
MSC	MultiShapeCreator

Chapter 1

Introduction

This thesis describes a prototype for an Interruption Management System (IMS) integrated into a development environment, to be used in the teaching of coding to children. It also presents an evaluation plan, but does not carry out the evaluation. To understand the motivation for, and structure of the thesis, we first introduce the appropriate ACM SIGSOFT research standards, and justify their use. Since one of the standards that will guide us is the Action Research standard, we next introduce the receptor organization we aim to help, STaBL Foundation.

1.1 ACM SIGSOFT Empirical Standards

The ACM SIGSOFT Empirical Standards¹ for Software Engineering are a type of evidence standard used to structure research such that ambitious empirical research standards can be met. Two of several standards are relevant to this thesis: (a) **Engineering Research Standard**, and (b) **Action Research Standard**. The

¹Available on <https://github.com/acmsigsoft/EmpiricalStandards>

Engineering Research (also known as Design Science) Standard is for research that invents and evaluates technological artifacts. As this thesis is in the Computer Science program, the term “Design Science” is more appropriate and will be used in place of “Engineering Research.” The Action Research standard covers empirical research that evaluates how the introduction of a tool affects a real-life context. Since one of the deliverables of this thesis will be a prototype, the Design Science Standard will be used as a scaffolding to structure the thesis, and to empirically evaluate the impact of the prototype on STaBL Foundation’s teaching.

The Action Research Standard will be used to develop a proposed evaluation to determine how the prototype will affect mentors in classrooms. Table 1.1 outlines each attribute in the Design Science Standard and the location of supporting material for that attribute in this thesis. According to Ralph et al. [43], “Design Science papers often define a motivating ‘problem’ or ‘objective’ rather than a research question.” As such, this thesis does not explicitly define research question(s) in lieu of a motivating problem. As a part of the Design Science methodology, the motivating problem is defined as a “How Might We?” (HMW) statement following conventional standards in Design Science. Siemon et al. [48] state that the “How Might We” method is often used to create a Design Challenge, which serves the development of an initial problem for Design Thinking, User-centered Design or even Design Science Research.” In the next section, the HMW statement is stated with necessary context about the motivating problem, and a detailed primer on Design Thinking is outlined in Chapter 3. But first, we must describe STaBL Foundation, the community our action is designed to change.

Design Science Standard

Attribute	Section
Describes the proposed artifact in adequate detail	Chapter 5
Justifies the need for, usefulness of, or relevance of the proposed artifact	Chapter 1, 2
Conceptually evaluates the proposed artifact	Chapter 6
Empirically evaluates the proposed artifact using Action Research	Chapter 3
Discusses state-of-art alternatives	Chapter 2, 3
Empirically compares the artifact to one or more state-of-the-art alternatives	Chapter 6
Assumptions (if any) are explicit, plausible and do not contradict each other or the contribution's goals	Chapter 1
Uses notation consistently (if any notation is used)	N/A - no notation is used in this thesis
Provides supplementary materials	Appendix A
Justifies any items missing from replication package	N/A - there are no items missing from the replication package
Discusses the theoretical basis of the artifact	Chapter 2

Provides correctness arguments for the key analytical and theoretical contributions	N/A - no theorems, complexity analyses, or mathematical proofs
Includes one or more running examples to elucidate the artifact	Chapter 5
Evaluates the artifact in an industry-relevant context (e.g. widely used open-source projects, professional programmers)	Chapter 4, 5

Table 1.1: Lists the Attributes that are required in the Design Science Standard and a reference to which section they are found in the thesis. A detailed list of the attributes and definitions for the Design Science standard are available in [43].

1.2 STaBL Foundation

STaBL Foundation² is a non-profit organization incorporated in Canada whose purpose is “to advance education by providing tools, curricula, and training in computer science and team-based innovation to children and teachers across Canada and around the world, especially those who are underrepresented in post-secondary computer science and software engineering programs.”

The STaBL Foundation has developed an Integrated-Development-Environment (IDE) to compile Elm code and display graphics to its users³. The aforementioned

²More information is available on <https://stablfoundation.org>

³Available on <https://www.STaBL.rocks>

IDE is known as STaBL.rocks and is used by many students across K-12 to build graphics and animations. A screenshot of the IDE is presented in Figure 1.1.

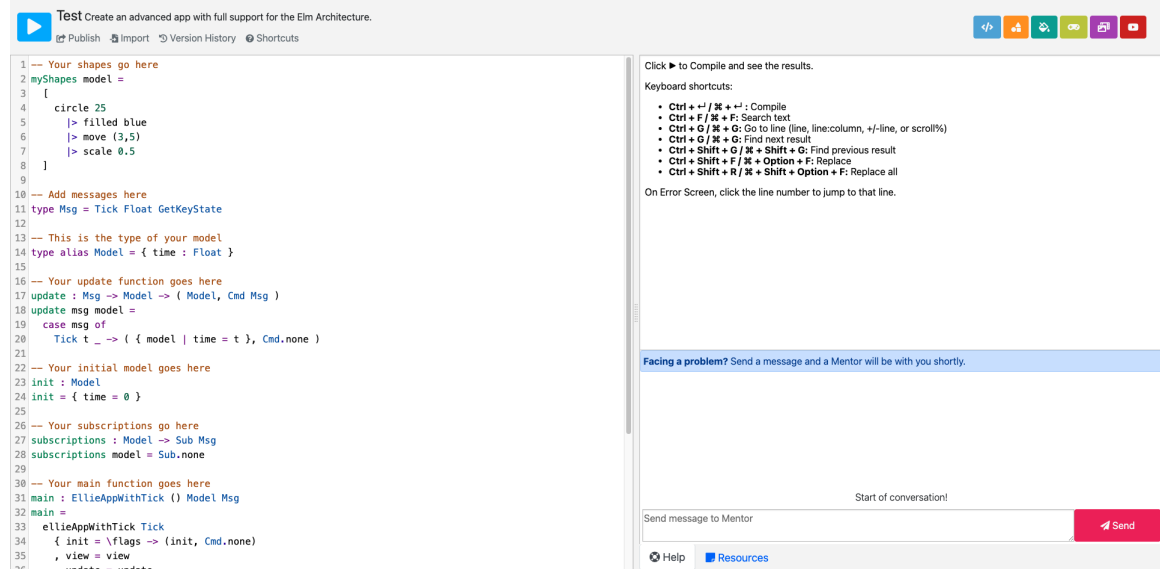


Figure 1.1: Screenshot of the STaBL.rocks IDE that is used by thousands of students to develop graphics and images. The left-hand side of the screen is where the user will write the code that is to be compiled, resulting in output on the top-right quadrant of the screen. Users have the ability to chat with a mentor as well, using the chat interface on the lower-right.

1.3 Motivating Problem

An innate feature of being a human being is the ability to understand when to interrupt someone such that you are not disturbing them. When humans are interacting with each other, they will typically understand the necessary social cues as to when to interrupt someone else. That is, humans typically know when to disturb someone, such as a coworker, to get their attention when they need the attention of another person for some reason; humans will gauge the other person to see how busy they

are by observing them before approaching them. Similarly, when the mentors in the STaBL Foundation go to classrooms to teach students Elm programming, they must also gauge whether a student is truly engaged in their task or not before interrupting them. If a student is engaged in a task and a teacher or mentor interrupts them then it can disrupt their cognitive processes and strip their focus away from the task at hand. If the student is interrupted at the right time, then the teacher or mentor can effectively collect the information that they require from the student without disrupting the student from their task. Unfortunately, effective interruption is difficult when the teacher/mentor to student ratio is low, or instruction is virtual. Typically there are two mentors per class of 20+ students when STaBL Foundation conducts workshops in classrooms and online. Naturally it is difficult for mentors to effectively interrupt students such that there is minimal disruption to their task.

All students must use the STaBL.rocks online coding platform when they are participating in an Elm workshop or learning session. If the platform itself is able to detect and prompt students at the optimal time of least resultant disruption, then the teachers and mentors would be further enabled to focus on other tasks. As such, we have developed a HMW statement that prompts the innovation required to develop a methodology and solution to interrupt students, such that there is the least amount of disruption to their primary task: **How might we interrupt students for announcements or to collect feedback, such that it results in the least amount of disruption to the interrupted student?**

1.4 Chapter Overview

The chapters in this thesis are organized such that a reader will learn about the existing literature on functional interruption management systems and then the necessary steps to develop a prototype of the aforementioned system. In **Chapter 2**, Literature Review, readers will learn about the origins of interruptions, why they are so disruptive, and common approaches to build software programs that are able to manage interruptions. **Chapter 3**, Methodology, presents an overview of the methodology that will be used to develop the interruption management system prototype by introducing the Design Thinking Methodology, followed by how it will be applied to build the prototype. The Action Research Standards will be applied here to develop a proposed evaluation method for the prototype to determine its usability for mentors. **Chapter 4**, Software Technology Stack, outlines the existing technology available to work with as a part of the STaBL Foundation, which will be the basis of the prototype. **Chapter 5**, Prototype, discusses the technical details of the prototype and how it was constructed, and an analysis of how interruptions will be managed in the prototype. Lastly, the Conclusion (**Chapter 6**) provides an overview of what was discovered and contributed, opportunities for novel contributions identified through gaps in the existing literature, as well as future work that may be completed to improve the prototype.

1.5 Contributions

This thesis discusses the design and implementation of an Interruption Management System (IMS) prototype, and its integration into two applications, Teacher Dashboard and MultiShapeCreator, in the STaBL Foundation. While the IMS prototype has been built specifically for the STaBL Foundation, it can be used in any Elm program which imports the IMS prototype’s accompanying library. The IMS prototype interrupts users in an “opportune” way by using a bounded deferral timer to defer interruptions to students until a maximum amount of time, which has been set by mentors, has passed. In the context of the IMS prototype, “opportune” interruptions are those that result in the least amount of disruption. The IMS prototype also uses Hierarchical Task Analysis (HTA) to identify “opportune” features that interrupt the student when they are clicked on. The IMS prototype has been integrated into the State-Diagram Draw application, discussed in Chapter 4, so that STaBL Foundation mentors are enabled to make announcements or collect feedback in Elm Workshops, discussed in Chapter 3, by interrupting students at an opportune moment.

Chapter 2

Literature Review

The scope of this literature review is to develop an understanding of the existing literature in the fields of interruption management systems and feedback collection mechanisms. The two fields will be studied to develop a software program, addressing the motivating problem outlined in the Introduction (Chapter 1).

To build a holistic understanding of how to design a software system that will result in minimal user interruptions, literature has been sourced from fields adjacent to Computer Science, such as Psychology, in addition to Computer Science. For example, the Seven Stages of Action framework developed by Norman and Miyata [35] was used as the basis of research for several studies relating to the development of interruption management systems [10, 11, 21, 22].

This literature review does not explore the methodology behind designing survey/questionnaire questions or educational psychology in collecting feedback during programming tasks. While interruptions have many implications across the fields of computer science, linguistics, and psychology [34], this thesis restricts the study of interruptions to the computer science domain.

The literature review is organized into three sections: **(1) Classification of Interruptions**, **(2) Disruptiveness of Interruptions**, and **(3) Interruption Management Systems**.

The first section, **Classification of Interruptions**, consists of an evaluation of the existing literature to outline several definitions and frameworks to systematically define and classify interruptions based on their resultant disruption levels. Further, there is a discussion on the different applications of each framework and when it is appropriate to use one framework over another. For example, McFarlane’s taxonomy [33] captures general information about disruption(s) resulting from an interruption, in contrast to Latorella’s Interruption Management State Model [25], which is used to capture more specific details about how an interruption disrupts a user from their primary task. The second section, **Disruptiveness of Interruptions**, outlines why interruptions are disruptive in the first place, and outlines the premise of developing interruption management systems. Lastly, the third section, **Interruption Management Systems**, outlines several approaches to develop software systems that aim to reduce an interruption’s resultant disruption to a user, also known as interruption management systems.

2.1 Classification of Interruptions

In this section, existing definitions of interruptions, and taxonomies and frameworks used to classify different types of interruptions are described. **Interruption Definition & Source** lists the existing definition of an interruption in the literature, and interruption categories describing different types of interruptions. **Interruption**

Classification Taxonomies evaluates several taxonomies and frameworks in the literature to classify different types of interruptions. Specifically, McFarlane’s taxonomy, Latorella’s Interruption Management State Model, and McCrickard’s Interruption-Reaction-Comprehension framework are described.

2.1.1 Interruption Definition & Source

There are several definitions of an interruption that exist in the current literature. However the two that are widely cited are: (a) McFarlane [33]’s unifying definition of an interruption, *“human interruption is the process of coordinating abrupt change in people’s activities.”*, and (b) McCrickard and Chewar [30]’s definition *“event prompting transition and reallocation of attention focus from a task to the notification.”*.

McFarlane’s work in 1997 aimed to develop a unifying definition and taxonomy of human interruptions. At the time, there were no existing unifying definition available, which limited researchers at the time to develop and integrate findings from other fields, build advanced taxonomies, and apply existing research. McFarlane attempted to bridge this gap in the existing literature by developing a proposal for the universal definition of an interruption.

McFarlane defines human interruption as a process, instead of a singular event, because he is attempting to capture all of the events surrounding the singular moment of interruption to a human, rather than just the singular moment alone. McFarlane’s holistic definition seeks to include how the interruption was generated, where is the human being interrupted, when are they being interrupted, why are they being interrupted, and what mechanism are they being interrupted by.

McCrickard also attempted to develop a definition of an interruption in 2003, and

in contrast to McFarlane’s definition, McCrickard defined human interruption as a singular event, because he was studying interruptions in the context of notification systems. McCrickard’s definition is limited to interruptions that are generated from a notification system, which can be from a computer, phone, etc., and are directed towards a human.

Aside from McFarlane’s and McCrickard’s definitions, there are also further distinctions with interruptions and how they affect the user that is being interrupted. For example, the study of interruptions in the field of Human-Computer-Interaction (HCI) typically occur between a computer system and a human and these types of interruptions can be further classified as *explicit and external* interruptions. They are **explicit and external interruptions** because they arise from a user’s physical or external environment and are designed to explicitly capture the user’s attention.

There are also internal interruptions that exist within a user’s mind in contrast to their external environment, and implicit interruptions whose purpose may not be to capture a user’s attention, however the scope of this thesis does not consider internal and implicit interruptions. The different types of interruptions are further defined in Table 2.3 for which detailed descriptions are available.

External and explicit human-computer interruptions, which are in the scope of this thesis, typically involve a computer system that generates interruptions within a desktop or mobile environment. Examples of external and explicit human-computer interruptions include a notification pop-up when a new message is received on a social media website, an ad pop-up when visiting a website, a notification ding when a new email is received, etc. Typical examples of a desktop environment are personal computers that a user will be using such as a laptop.

Interruption Category	Definition
Internal Interruptions	Occur due to our own background thought process.
External Interruptions	Result from events in the physical environment.
Implicit Interruptions	Arrive from a process that belongs to the ongoing task.
Explicit Interruptions	Arrive from a process that does not belong to the ongoing task.
Human-to-human	A human interrupts another human resulting in their attention being taken away from some primary task.
Human-to-computer	A computer system interrupts a human by drawing their attention to the computer system or a process within the computer system, away from the human's primary task.

Table 2.1: Categories of interruptions.

Mobile environments typically consist of handheld devices such as a tablet (iPad, Kindle, etc.) or a phone (iPhone, Android phone, etc.)

Desktop and mobile environments differ in the amount of readily available information used to detect when to effectively interrupt a user. Desktop environments contain less readily available information since they are less portable than a mobile device. Subsequently, desktop web browsers contain even less readily available information than desktop applications, which implies that there is less data available to build categories for interruptions and subsequently interruption management systems [34].

While building software systems for desktop web applications, external interruptions that result from a system to a human and in the context of a desktop web application are to be considered more than the other types of interruptions.

2.1.2 Interruption Classification Taxonomies

The existing literature to classify an interruption in the context of external and explicit human-computer interruptions, has generally identified three frameworks: (a) McFarlane’s Taxonomy, (b) Interruption State Management Model, (c) Interruption-Reaction-Comprehension Framework. Each of these existing frameworks offers a novel way to classify interruptions based on different characteristics about each interruption. McFarlane’s taxonomy focuses on eight broad dimensions outlining a holistic description of the interruption, such as the source of the interruption, and characteristics of the user being interrupted [33]. The Interruption State Management Model builds on-top of McFarlane’s taxonomy, which outlines an interruption in more detail than McFarlane’s Taxonomy, by offering a classification system [25]. McFarlane developed a taxonomy for interruptions and Latorella developed the Interruption Management State Model (IMSM) to map interruption management resolutions to discrete steps and their related consequences. The IMSM can be jointly used with McFarlane’s taxonomy to develop a detailed dictionary of how a particular interruption will have intended or unintended consequences. Lastly, the Interruption-Reaction-Comprehension (IRC) framework uses the three dimensions, interruption, reaction, and comprehension in regards to a user’s perception of the interruption to classify various interruptions [31].

McFarlane’s Taxonomy

Ultimately, not all interruptions are the same and a taxonomy was necessary to determine the classification and subsequent consequences of different types of interruptions. Significant work has been completed by McFarlane and Latorella in order to

organize interruptions into a taxonomy and define frameworks to capture how a user will respond to an interruption. McFarlane’s Taxonomy was founded on an extensive body of existing theoretical research in User Interface (UI) design. McFarlane [33] identified eight dimensions to classify interruptions:

- **Source of Interruption:** Captures from whom or what the interruption originated (e.g., self, another person, computer, etc.).
- **Characteristics of the User Being Interrupted:** Captures the ability of an individual to process an interruption and individual differences (e.g., personality, motivations, etc.)
- **Coordination Method:** Captures how reactive a user has to be to respond to an interruption. There are four primary design solutions: (a) immediate interruption, (b) negotiated interruption, (c) mediated interruption, and (d) scheduled interruption.
- **Interruption Meaning:** Captures the purpose of the interruption and why it needs to take place (e.g., alert, attack, distribute attention, etc.).
- **Expression Method:** Captures how the interruption will be presented to the user (tone, expression, physical/virtual, etc.).
- **Channel of Conveyance:** Captures the channel of communication used to deliver the interruption (e.g., face-to-face, human-to-system, etc.).
- **Human Activity Changed by Interruptions:** Captures how users interact or act differently after the interruption has been delivered (e.g., individual activities, internal, external, etc.).

- **Interruption Effect:** Captures any performance changes on a primary task that may occur as a result of the interruption (e.g., change in human activity, change in focus, change in social relationships, etc.).

Additionally, McFarlane’s Taxonomy identifies four primary design solutions to resolve user interruption, which are the four Coordination Methods mentioned in the “Coordination Method” dimension of McFarlane’s taxonomy:

- **Immediate Interruption:** Interrupts the user in a way that demands that the user stops what they are doing and immediately tend to the interruption.
- **Negotiated Interruption:** Interrupts the user in a way where the user has the option to tend to the interruption immediately or postpone to a later time.
- **Mediated Interruption:** Interrupts the user’s proxy, and the proxy decides how the user should be interrupted, depending on the importance of the interruption to the user.
- **Scheduled Interruption:** Interrupts the user on a scheduled basis (e.g., once every 15 minutes).

These four primary design solutions can be further illustrated through an example of an employee working on their work computer and an enterprise chat system that needs to initiate an interaction with the user. An *immediate* solution would result in a pop-up on the worker’s screen that they cannot close and must tend to. A *negotiated* solution would be a pop-up on the worker’s screen that they can tend to, minimize or defer to another time. A *mediated* solution would be the enterprise chat system prioritizing the interruption it needs to deliver to the user, and then selecting an

immediate or negotiated solution. Lastly, a *scheduled* solution would be an attempt to interrupt the employee once every 5 minutes.

McFarlane’s study also identified that negotiation-based solutions appear to generate the best results. Negotiated solutions are the most common in a human-to-human interaction where the individual being interrupted would have the option to: (a) handle the interruption immediately, (b) accept the task of handling the interruption and do it at a later date, (c) refuse to handle the interruption, or (d) ignore the interruption.

Interruption Management State Model

Latorella [25] worked on developing Interruption Management State Model (IMSM) to model how a user would react to an interruption with more granularity than McFarlane’s Taxonomy. That is, given a particular type of interruption, the IMSM identifies the subsequent consequences of that interruption to the user. When there are enough commonalities with how various interruptions affect users, they can be grouped and classified together. The rest of this subsection further details the IMSM, and a detailed visualization is captured in Figure 2.1.

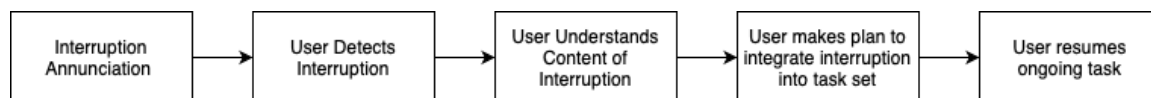


Figure 2.1: Interruption Management State Model (IMSM) discrete steps in completing an ongoing task with an interruption [32].

- **Detection of the Interruption Annunciation:** Each interruption will necessarily announce its arrival to a user, and this step captures when a user first notices the interruption.

- **Interpretation of the Annunciation:** A user will need to understand what information the interruption is requiring or delivering.
- **Integration of the interruption into the ongoing task set:** A user will need to organize their planned work so that they can effectively respond to the interruption and merge or integrate it into their existing task set or work flow.
- **Resumption of the ongoing task set:** Once the interruption has been handled, then the user will need to resume their ongoing tasks or work.

The model also defines four general effects of interruptions in terms of the information processing stages.

- **Diversion:** Occurs when an interruption redirects the attention of a user from a primary task to the interruption.
- **Distraction:** Occurs when a momentary diversion takes place.
- **Disturbance:** Occurs when the primary task is placed on hold to tend to the interruption source to immediately tend to the interruption or delay it to a later time.
- **Disruption:** *Diversion, Distraction, and Disturbance* may lead to lower downstream primary task performance resulting in Disruption.

The IMSM identifies five distinct responses to an interruption:

- **Oblivious Dismissal:** A user does not detect the interruption and it is missed.

- **Unintentional Dismissal:** A user accidentally dismisses the interruption because they misinterpreted the interruption.
- **Intentional Dismissal:** A user correctly interprets the interrupting task but decides not to perform the required interrupting task.
- **Preemptive Integration:** A user correctly interprets the interrupting task and immediately performs it.
- **Intentional Integration:** A user integrates the interrupting task into the existing primary task set, and the user rationally decides when to complete the interrupting task.

There are also three phases identified during a user's task-switch between a primary task and an interrupting task:

- **Before Switch:** Occurs before the user switches between the primary task and the interrupting task.
- **During Switch:** Occurs while the user is addressing the interrupting task.
- **After Switch:** Occurs after the user has addressed the interrupting task.

Interruption-Reaction-Comprehension (IRC) Framework

McCrickard et al. [31] recognized three critical dimensions for modeling of notification system user goals by developing the Interruption-Reaction-Comprehension (IRC) framework; the IRC framework consists of three dimensions: (1) Interruption, (2) Reaction, and (3) Comprehension. A visual representation of the three aforementioned

dimensions is presented in Figure 2.2. The individual dimensions of the IRC framework can be defined as follows:

- **Interruption:** “An event prompting transition and reallocation of attention focus from a task to the notification”
- **Reaction:** “The rapid and accurate response to the stimuli provided by notification systems”
- **Comprehension:** “Using notification systems with the goal of remembering and making sense of the information they convey at a later time.”

The IRC framework only considers a boolean value when attempting to classify notification types, assigning a 0 or 1 value to each of the dimensions. A value of 0 indicates a “low” value for a particular dimension whereas a value of 1 indicates a “high” value, and there must be a value assigned to each dimension in the IRC framework. For example, if the goal is to provide a notification that results in low interruption, high reaction, and low comprehension then a chat pop-up can be used, which rapidly captures the user’s attention but allows the user to attend to the notification when they would like to. An example of low interruption, low reaction, and low comprehension would be white noise that does not disturb a user at all, such as a fan operating harmlessly in the background. An example of high interruption, high reaction, and high comprehension would be a critical error pop-up, which immediately demands the attention of a user, where the user must attend to the pop-up before moving on with any other task that they were engaged in. Several other examples, outlining the combinations of the three different dimensions in the IRC Framework, are detailed in Table 2.2.

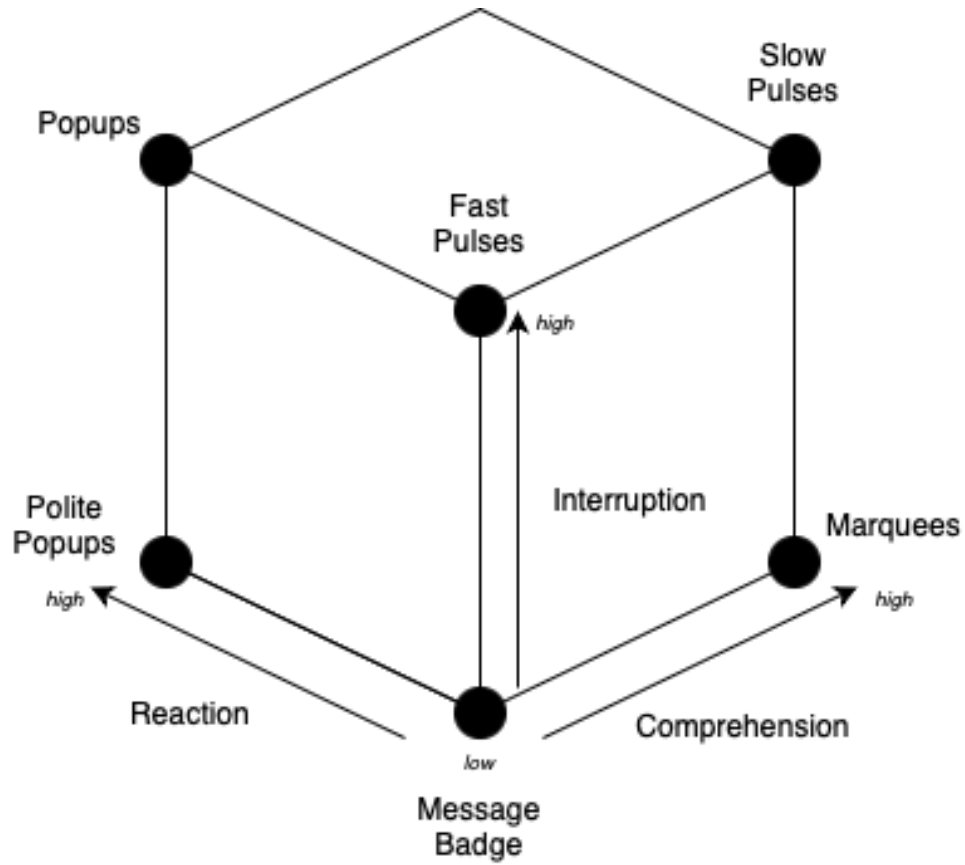


Figure 2.2: Interruption, Reaction, Comprehension (IRC) framework with low or high values for each dimension [31]

2.2 Disruptiveness of Interruptions

Some interruptions are more disruptive than other interruptions, and there has been significant work done in the field of Human-Computer-Interaction (HCI) to understand the “disruptiveness” of certain types of interruptions. Several studies have concluded that disruptions delivered at inopportune moments to a user typically result in degrading performance on the primary task that they were interrupted from [11, 18, 36]. The only notable exception exists in the experiments conducted by Lewin [28] who concluded that interruptions can improve the performance of a primary task.

I	R	C	Example
0	0	0	Noise
0	0	1	Scrolling Marquee
0	1	0	Chat Pop-up
0	1	1	Message Badge
1	0	0	Alarm
1	0	1	Polite Pop-up
1	1	0	Warning Pop-up
1	1	1	Critical Error Pop-up

Table 2.2: Examples of the Interruption (I), Reaction (R), Comprehension (C) framework with low (0) or high (1) values to capture blending of dimensions. Willermark et al. [50] have developed the polite pop-up, which is a “modal pop-up that is created based on click events.” This allows the user to choose to interact with the pop-up rather than automatically placing a pop-up on-top of the users’ display like a traditional pop-up.

Their work did not measure actual task performance, but instead measured a participant’s ability to recall the nature of the task when they were interrupted. In contrast, the empirical studies in support of the conclusion that interruptions typically have a negative effect on primary task performance had their users engage in some measured task where performance was tracked with the presence of interruptions. Generally, the effect of an interruption on a participant’s primary task performance is predicted by the user’s cognitive load at the time of the interruption [35], and the relevance of the interruption to the task [11].

Miyata and Norman [36] studied the process of detecting high cognitive load in a participant by trying to understand how the participant is thinking or reasoning about a task. Specifically, they investigated cognitive load while the participants were completing a task, before they started the task, and after they completed the task. Jointly the two developed a framework, titled the Seven Stages of Action, consisting

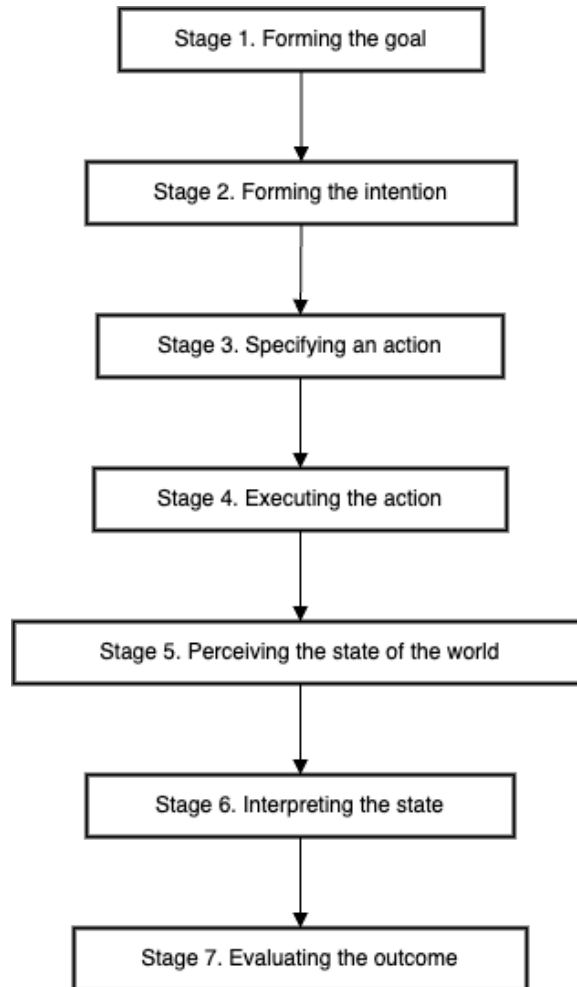


Figure 2.3: The Seven Stages of Action outlined by Norman and Miyata. The least amount of disruption to a user completing a task will occur during the formation of a new task, from Stage #7 to Stage #1 or in between Stages.[36]

of seven stages to outline the steps a user must take to complete and task.

Figure 2.3 outlines the Seven Stages of Action with each sequential step organized in a flow chart. The first four stages (a through d) are considered the Gulf of Execution and the last three stages (e through g) are considered to be the Gulf of Evaluation. The Gulf of Execution reflects the processes that the user has to engage in to bridge

the gap between forming an intention and executing actions to complete a goal. Whereas, the Gulf of Evaluation reflects the processes that the user must engage in to interpret the results of their actions and whether the goal that they had set has been met. When a user is engaging in a task, the first step that they will take is to Form the Goal, where the user forms a goal about something that they want to accomplish. Next they will develop an intention to act on the goal that they have set in their mind. After that they will specify actions that need to be completed in order to complete their goal; a sequence of actions that will lead the user to their goal. Once the user has executed on the actions, they will then begin to perceive the results of their actions in the world. This perception leads to the interpretation and understanding of whether their actions have resulted in satisfying the goal or not. Lastly the user will evaluate whether the goal has been reached or not.

A user that is completing a task will typically have a high cognitive load when the information required to complete the task has not been externalized, that is the information to complete the task resides in the user's mind. Norman and Miyata concluded that the least amount of cognitive or memory load would be when a user has completed an action and is forming the goal of the new action. Cognitive or memory load is the highest during the stages of the formation of the intention, specification of the action, perceiving and interpreting the state of the world, and evaluating the outcome, because information is highly internalized during these stages.

Czerwinski and Horvitz [10] investigated the effects of Instant Messaging (IM) notifications during an activity on the desktop computer. The activity involved managing a basket of stocks with a strike price, and were told that they would be notified of an opportune time to sell these stocks for a higher price, resulting in a profit. With

this context, the participants were then instructed to complete a set of activities using common desktop tools (Microsoft Word, Excel, etc.) and would be interrupted with notifications about stock prices while they were completing these tasks. Notifications arriving during Stages 2 - 6 (inclusive) of the Seven Stages of Action resulted in more disruption compared to notifications received during Stage 1 or Stage 7; captured by the time it took to complete a task. More simply, participants were more likely to take longer to complete a task when they were deeply engaged with the task. Specifically, sending a notification when users were typing or interacting with toolbars was reported to have resulted in significantly more disruption. Czerwinski and Horvitz [10] further note that longer task completion times may be attributable to delays resulting from visual reorienting after reading a notification, or latency associated with the users' memory as they try to recall their primary task [11]. Bailey et al. [5] conducted an experiment to validate why interrupting a user's task should be avoided by measuring the effects of an interruption on a user's task performance, annoyance, and anxiety. Results demonstrate that an interruption has a disruptive effect on both a user's task performance and emotional state, and the degree of disruption depends on the user's mental load at the point of interruption [5]. A notable exception in the literature is the experiments completed by Gillie and Broadbent [18] where they concluded that a higher working memory load is not necessarily correlated with how disruptive an interruption is, however later experiments conducted by the same researchers with more complex interruptions did result in the respective correlation.

2.3 Interruption Management Systems

This section describes a common Interruption Management System (IMS) architecture design, the underlying mechanisms to determine when the IMS should deliver an interruption, and the benefits of the personification of software systems directed towards children. The personification of software systems is necessary in this section because the motivating problem, stated in Chapter 1, requires the solution to be developed for young students, or children, using the STaBL.rocks integrated development environment (IDE). A comprehensive overview of the STaBL.rocks IDE is provided in Chapter 4.

2.3.1 Architecture

The key objective of an Interruption Management System (IMS) is to help users to effectively perform their primary task and make computing devices calm by unobtrusively mediating interruptions [22]. IMSs are limited within the web browser on a desktop environment because of the limited information available about the user. As such, generally, in a desktop web browser environment an IMS can be broken down into three major components in order to facilitate the aforementioned service: (a) *input data sensor*, which collects data primarily through the keyboard and mouse, (b) *interruption model*, which determines whether a user should be interrupted or not, and (c) *delivery system*, which determines how the user should be interrupted [34]. Furthermore, the IMS as a whole can generally be developed by: (a) *collecting input data*, (b) *constructing features*, and (c) *model training*. One development paradigm that has emerged from the body of literature is to use an IMS to exploit task phases of a user, so that interruptions are the least disruptive when delivered.

2.3.2 Exploiting Task Phases

The paradigm of exploiting task phases when building an IMS is founded on Norman and Miyata's research of the Seven Stages of Action. Norman and Miyata argued that when a user is completing a task, there will be certain stages where they will be less prone to a disruption from an interruption than another stage. The Exploiting Task Phases paradigm attempts to predict when a user is in a stage where an interruption will not be disruptive, and deliver an interruption or notification at that time.

Czerwinski et al. [11] identified in their study that users typically will take longer to complete their primary task when they receive an interruption in the execution phase. Further, users typically took longer to switch between the primary task and the interruption task during the execution phase, and when the interrupting task was not related to the primary task. They theorized that users will not switch immediately to the interrupting task because they are planning on switching to the interrupting task upon completion of a sub-task. They further support Norman and Miyata's claims because their study concluded that once a user has become deeply engaged in a task goal, notifications will be disruptive; the best time to deliver notifications is early in the task, before the user is engrossed by the task [11]. Cutrell et al. [9] also investigated Norman and Miyata's claims of the disruptiveness of interruptions during task completion, and similarly to Czerwinski and Horvitz [10], they also concluded that interruptions, from notifications, were generally disruptive when a user was engaged in a task. However, they also claimed that delivering notifications early into a task had a higher chance of the user forgetting their primary task, compared to interruptions that were delivered later on [10]. Both studies identified that delaying the delivery of interruptions via notifications until an opportune moment, resulted in

better primary task performance.

The idea of delaying or deferring an interruption to a later time was formalized by Horvitz et al. [21] who suggested to delay an interruption if it was not being delivered at an opportune moment up until a maximum deferral time had been reached, which they termed *Bounded Deferral*. For example, if the interruption model determined that interrupting the user immediately would result in disruption, then the delivery system would wait until a predefined time to deliver the interruption, if the model didn't already send the message to deliver the interruption.

There have also been other experiments that use peripheral sensors to determine which task phase a user is in such as the experiments conducted by Iqbal et al. [22] who used ambient sensors, such as cameras to observe pupillary response, to determine when a user was experiencing high cognitive load. Pupil size was correlated with cognitive load and then mapped to specific features and their usage within an application, such as clicking a toolbar, opening a document, etc. [22]. In their application, they then started to deliver notifications when a user was not experiencing high cognitive load according to historical feature mapping data.

2.3.3 Identifying Task Phases

Hierarchical Task Analysis (HTA) is a technique that can be used to objectively create tasks, sub-tasks and user goals, which can then be used to evaluate when a user is in between a sub-task boundary or a task boundary. Annett [2] first proposed HTA to recognize training requirements for specific tasks in a factory or power plant setting; however, it can be leveraged and applied to any process such that the process has an end goal [15]. Applying HTA requires an identification of a goal, followed by tasks and

Table 2.3: Experiment types in the area of interruption management systems in a desktop environment.

Determining Task Breakpoints	Exploiting task breakpoints to not deliver interruptions at inopportune moments, and to delay the delivery of interruptions until an opportune moment can be found. [11, 10]
Deference of Interruptions	Deferring interruptions until some maximum allotted time to deliver interruptions at opportune moments. [21]
Inferring Cognitive Load	Correlating pupil size and cognitive load to determine which feature sets result in low cognitive usage. [22]

sub-tasks that are required to fulfill that goal. There can also be the identification of sub-goals that are a part of a larger goal, which also follow the same structure of having tasks and sub-tasks as a requirement of completion. For example, if a user needs to complete a task on a web application, then HTA will inform the sub-tasks that they need to complete to meet their goal of completing the task. Identification of goals, their tasks, and sub-tasks through HTA creates an opportunity to understand when a user is engaged in a task and when they're not, by mapping them to the Seven Stages of Action designed by Norman and Miyata.

2.3.4 Delivery System Personification

Typically, novice programmers will experience a lot of rejection and failure when they first start coding, especially with software platforms that allow free-form text to be entered. Compilation and runtime errors can often be cryptic, and the one-sided communication from the system can be negatively interpreted [23]. Interacting with a personified software agent results in deeper task engagement as concluded by Walker

et al. [49] who compared subjects' responses to an interview survey under three conditions where questions were posed with a synthesized voice: (a) neutral expression, (b) stern expression, (c) text-only. Participants in the experiment were more responsive with the synthesized voice rather than the text-only approach [49]. Lee and Ko [26] developed a personified software agent named Gidget that is a personified robot feedback collector. Participants engaged with the tool, which Gidget was hosted on, for an average time of 50.1 minutes compared to 39.4 minutes of engagement time for participants that didn't interact with Gidget [26]. The personification of an interruption delivery system necessarily will have "human-like" features, such as a face that will have expressions semantically relating to the state of the system; for example, a happy face when a "successful" event occurs such as a successful code compilation or a sad face when there is a compilation error.

Chapter 3

Methodology

The methodology section of this thesis focuses on first, describing the basics of “Design Thinking”, followed by the design of the research study. Design Thinking will be used, as a tool, to develop a usable prototype and the research study design will be used as a scaffolding to evaluate the usability of the prototype. The research study design follows the Empirical Standards Framework’s Design Science Standard, described in Chapter 1, where the Action Research standard is used to further define a proposed empirical evaluation method which meets the defined standard.

3.1 Design Thinking Primer

Design Thinking is a solution-oriented framework originating from Rowe’s book, “Design Thinking”, in 1987 [45]. It is used in the field of software development to develop user requirements using tools such as Mural or Figma, and is well suited for software development because iterations are rapid. However, Design Thinking as a tool can be applied to pretty much any field. In the field of Design Thinking, two popular

frameworks have emerged which further enable the user of the framework to deliver the right (software) product; the two frameworks are: (a) The Five Stages of the Design Thinking and (b) the Double Diamond framework. Each framework is explored in detail in this primer to capture the differences between each approach. The broad pattern that each framework must follow of Design Thinking is that it is typically a non-linear process where each stage does not necessarily have to be followed in order, but rather to the needs of the user. As such, certain processes will be restarted (iterated on) once new information is learned so that the customer or user of the developed solution is enabled with the most useful product.

According to Interaction Design Foundation, the Five Stages of Design Thinking are as follows, according to the Interaction Design Foundation [13]:

1. **Empathize:** Develop an understanding of the user’s needs, using Empathy, to identify problems and pain points that the user may be facing.
2. **Define:** Distill the information that was provided by the user in the Empathize phase to create insights. Typically, a problem statement is a deliverable of this phase. A common example of a problem statement is one of the format, “How might we {STATE MAIN PROBLEM} so that {MAIN RESOLUTION OF PROBLEM}?”
3. **Ideate:** Brainstorm solutions for the originating problem statements from the Define phase. Try to approach the problem from as many different angles as possible to develop a diverse set of solutions. Solutions must then be organized by whether they solve the problem identified in the problem statement, and how feasible the solution is.

4. **Prototype:** Low fidelity prototypes must be developed using a few solutions to present and interact with the users. Low fidelity prototypes are necessary because iterations can be developed quickly, rather than waiting for a long time to implement a solution in a high fidelity prototype.
5. **Test:** Test the low fidelity prototypes developed in the Prototype phase with the user pool to determine whether it is truly solve their problems or not. Collect feedback, and use the feedback to iterate and develop a better solution by restarting any of the five stages with this new information.

While the five stages are listed in a sequential order, it is always possible (and encouraged) to go back to a previous stage in the process when new information is learned that necessitates going back to a previous stage. For example, if a product development team is in the Ideate phase but they learn about a new problem that their user is having, then they should go back to the Empathize phase to learn more about their users' problems before developing a solution. The non-linear nature of the Five Stages of Design Thinking and the overall process itself, are further illustrated in Figure 3.1.

The Double Diamond is a design thinking framework that was developed in 2004 and is currently maintained by the Design Council. The aim of the framework is to systematize the Design Thinking process to develop a solution for a group of users. Broadly, the framework implores its users to initially diverge to discover as many problems or pain points as possible, converge to a subset of root causes, diverge to brainstorm as many possible solutions as possible, and to converge on a solution which addresses as many of the root causes as possible. The framework can be applied in an iterative manner, so that a more optimal solution can be continuously delivered

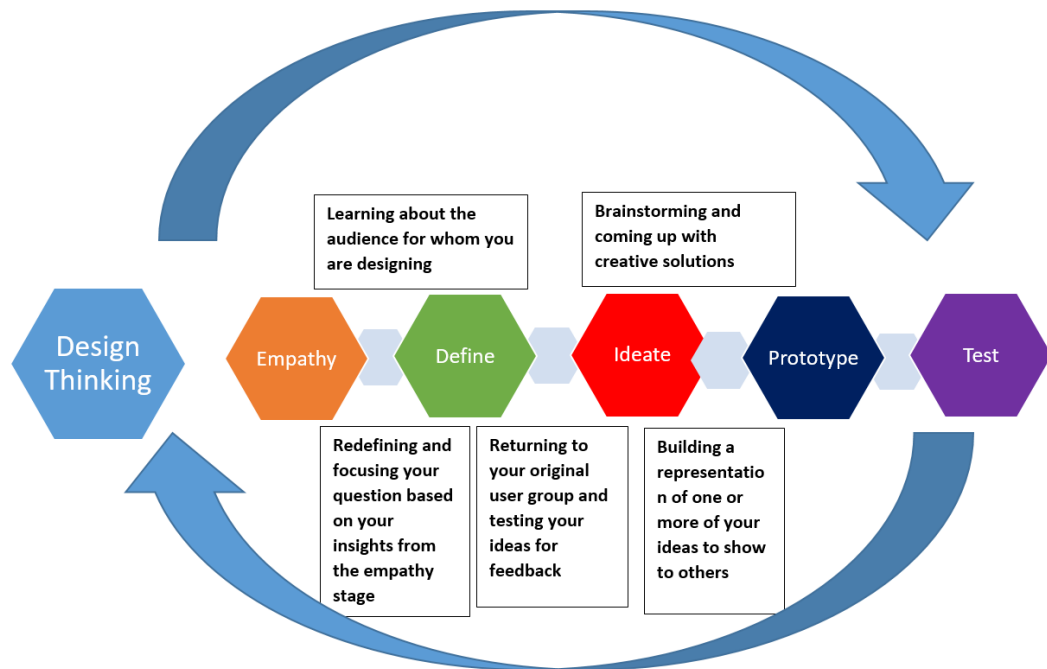


Figure 3.1: An illustration of the Five Stages of Design Thinking process, where the five steps are: (1) Empathize, (2) Define, (3) Ideate, (4) Prototype, and (5) Test. At any point, a user of this process may go back to a previous step and start over again to deliver the best product to the user, capturing the “iterative” nature of this process. [20].

when more information is learned about a product’s users [17]. A visualization of the Double Diamond framework is in Figure 3.2. The Double Diamond consists of four main steps:

1. **Discover:** Collect as much information as possible by speaking with those that are affected by the problem rather than assuming what the problem is.
2. **Define:** Collate and group the information from the Discover phase to create data and insights; clearly define the problem.
3. **Develop:** Brainstorm solutions to the clearly defined problem from the Define stage.
4. **Deliver:** Deliver solutions on a small-scale and iterate using feedback collected so that solutions can be improved.

These four steps are also associated with four design principles which add flavour to each step:

- **Put people first:** Begin with understanding who the users are, and what problems they face.
- **Communicate visually and inclusively:** Users should be brought along the solution development process by developing a shared understanding of the problem.
- **Collaborate and co-create:** Source other opinions to holistically develop a strong solution.
- **Iterate, iterate, iterate:** Collect feedback on small prototypes to understand where optimizations can be made right away.

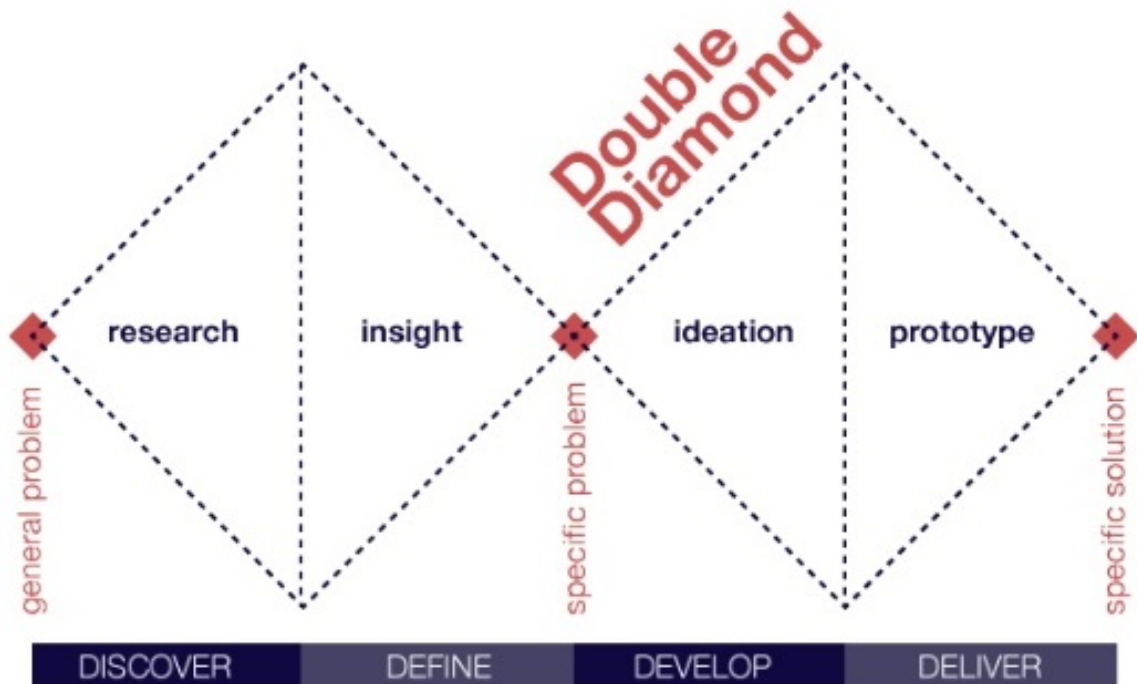


Figure 3.2: The Double Diamond Framework outlining the four distinct and iterative stages: (a) Discover, (b) Define, (c) Develop, and (d) Deliver.

3.2 Designing the First Iteration of the Prototype

The motivating problem, stated in Chapter 1, exists at the intersection of the two diamonds in the Double Diamond framework. Specifically, it is in between the “Define” and “Develop” stages, where a “specific problem” is defined. To ideate solutions for the first iteration of the prototype, an ideation session was organized with the Managing Director (MD) of the STaBL Foundation. The MD, over the last ten years, has spent many hours visiting K-12 classrooms, instructing mentors, and building the STaBL Foundation’s technology platform, STaBL.rocks. The MD is able to distill insights from many hours of conversation and instructional experience with mentors into features for an initial prototype, which is necessary to ideate solutions for the motivating problem.

During the ideation session, the outcomes of the literature review, in Chapter 2, were discussed with the MD to outline the potential components of a solution to deliver notifications to students. The following items were highlighted:

- Interruptions and notifications must be delivered so that they do not interrupt the student as much as possible. The *Bounded Deferral* [21] approach by Horvitz et al. must be used so that notifications are delivered before the end of class.
- Teachers or admins must be able to trigger notifications to urgently deliver information, such as the end of a class.
- Features within the IDE will be tagged with different phases of Norman and Miyata’s Seven Stages of Action, so that the Interruption Management System will trigger notifications at opportune moments [36].

- McFarlane and Latorella [32]’s Taxonomy, Latorella [25]’s Interruption Management State Model Taxonomy, and McCrickard et al. [31]’s Interruption-Reaction-Comprehension Framework were discussed to determine which taxonomy could be implemented with the prototype.

The outcome of the ideation session is a list of requirements that are necessary to deliver a functional and usable IMS prototype. The following set of requirements, which are also present in Table 3.1, must be present for the solution to be usable:

1. Must be integrable into the MultiShapeCreator and the Teacher’s Dashboard. MultiShapeCreator and the Teacher’s Dashboard are described in Chapter 4.
2. Must be reusable by other applications built in the STaBL.rocks IDE, such as State-Diagram Draw. State-Diagram Draw is discussed in Chapter 4.
3. Must be able to support custom functions, such as the output of a machine learning model, to trigger notifications.
4. Must contain a personified software tool to deliver notifications to the user.
5. Must contain a component library which other developers can use to build interruption management systems in STaBL.rocks.
6. Must contain documentation which can be used to capture McFarlane’s Taxonomy. It was decided to use McFarlane’s Taxonomy instead of Latorella’s IMSM because observation and documentation of individual users is required to develop the IMSM Taxonomy, whereas it is not for McFarlane’s Taxonomy. Latorella’s IMSM Taxonomy can be further developed once the Interruption

ID	Description
1	Must be integrable the MultiShapeCreator
2	Must be integrable into the Teacher’s Dashboard
3	Reusable by other applications in STaBL.rocks
4	Support custom functions to trigger notifications
5	Component library to build interruption management systems
6	McFarlane’s Taxonomy Documentation
7	Tag components with Seven Stages of Action
8	All interruptions must abide to Bounded Deferral

Table 3.1: Requirements table for developing the initial prototype for the STaBL Rocks Interruption Management System.

Management System (IMS) prototype has been evaluated using the proposed plan in Section 3.3.

7. Must provide the capacity to tag individual or groups of UI elements with Norman and Miyata’s Seven Stages of Action to support Hierarchical Task Analysis. Hierarchical Task Analysis is discussed in Chapter 2.

3.2.1 Design Methodology

The Five Stages of Design Thinking and Double Diamond frameworks will be heavily leveraged to collect feedback in this study. Specifically, materials will be developed to collect feedback from mentors that are currently employed or are volunteering with STaBL Foundation. Visualization tools such as MURAL will be used to document any findings from each of the stages in the Design Thinking framework.

The first iteration of the prototype will be developed using the three components of an Interruption Management System (IMS), defined in the Literature Review: (a) Sensor, (b) Interruption Model, (c) Delivery System. Additionally, there will be

sub-components available for each respective component, which can be used to build different versions of each component. For example, the Interruption Model component may consist of a machine learning model to exploit task phases, or a dictionary of features mapped to task phases. Ultimately, each feature will be categorized under one of the three major architecture components to further illustrate the inner workings of the system.

3.2.2 Collecting Feedback to Inform Design Iterations

In the Design Thinking process, once a prototype has been developed, feedback must be collected from users, who are mentors and students, to evaluate the usefulness of the solution, and to inform the next design iteration of the prototype. It was decided to not interview students because interviewing K-12 students introduces ethical and logistical complexities with consent and data quality, especially with younger students, whereas that is not the case for mentors.

Several studies have used adults, in-place of students, in the design process of a tool [1, 7, 16, 24, 27, 29, 44]. Adults, who can be teachers or mentors, are proxies for students that transmit their agency and represent their best interests. Rhode Island School of Design and Leighton Beaman [44] define proxies as “stand-ins for another - a person, an organization, an action or a process.” As such, mentors are stand-ins or proxies for students in the design process of the prototype, and will be interviewed in-place of students.

Mentors were chosen to be proxies for students because mentors have spent many hours in classrooms understanding the pain points of many different types of students. A group of mentors offers many different points of view, reflective of the different

interactions with K-12 students. Ultimately, it is a lot more efficient to interview mentors until no more meaningful feedback can be collected from them to inform the next iteration of the prototype.

The MD and several mentors were invited to participate in semi-structured interviews. A semi-structured interview format is necessary, over other interview formats, because, while feedback must be collected for a specific set of features (structured), there must be an opportunity for open-dialogue to understand any deeper concerns that the interviewee has.

The semi-structured interview consisted of five questions:

1. Can you tell me a little bit about yourself and your mentoring experience?
2. Do you think that the personified tool is effective at capturing a user's attention?
Why or why not?
3. Do you think mentors would use this tool to collect feedback from students?
Why or why not?
4. Do you think this tool can be integrated into other STaBL.rocks applications?
5. If you had a magic wand, what would you change about this tool?

The first question is intended to develop rapport with the interviewee, whereas the last question is intended to be a catch-all question. A catch-all question invites the interviewee to contribute more useful information that may not have been elicited by the interviewer in previous questions.

The other three questions were developed by sourcing similar statements from an existing Design Thinking Question Bank, which is available in A. The sourced statements are as follows:

1. I found this ⟨Application X⟩ confusing to use.
2. I could not do some of the things I needed to do while using ⟨Application X⟩.
3. Using ⟨Application X⟩ was worthwhile.
4. I would recommend this ⟨Application X⟩ to other mentors
5. I felt interested in the ⟨Application X⟩.

The aforementioned statements from the Design Thinking Question Bank were specifically chosen to capture the mentors’ experiences while using the Interruption Management System (IMS) prototype. Other statements in the question bank were repetitive, of the ones listed, or did not pertain to collecting feedback for an application.

It was decided to not use the statements from the question bank directly because it would be more difficult to facilitate open dialogue with the interviewees if they were asked to respond to direct statements. Instead, it was decided to reframe the statements as questions to promote open dialogue, using follow-up questions, about the usability of the prototype.

The semi-structured interviews were virtual, due to geographical limitations, and 2-3 mentors and the MD were interviewed for each design iteration. After each interview cycle, thematic analysis was applied to design prototype features from the interviewees’ responses. Braun et al. [6] describe thematic analysis as a “method for identifying, analyzing, and reporting patterns (themes) within data.” The purpose of the thematic analysis is to identify common features, through patterns in the interview transcripts and/or notes. Specifically, an inductive analysis approach was applied by coding transcripts, extracting themes, and mapping themes to requirements for the

next design iteration. An inductive analysis approach strongly maps identified themes to the transcripts. Inductive analysis is necessary because the next iteration of the prototype will be informed from the themes identified from the data, and there is no pre-existing theory which can be used to perform a deductive analysis.

3.3 Research Study Design

Table 3.2: Attributes that are required in the Action Research Standards and a reference to which section they are found in the thesis. Some attributes are not applicable, marked using “N/A”, because an empirical study, which has not been performed in this thesis, is required to determine the necessary information for the attributes.

Action Research Standards	
Attribute	Section #
Justifies the selection of the site(s) that was(were) studied	3.3.2
Describes the site(s) in rich detail	3.3.2
Describes the relationship between the researcher and the host organization	3.3.1
Describes the intervention(s) in detail	3.3.3
Describes how interventions were determined	3.3.3
Explains how the interventions are evaluated	3.3.3
Describes the longitudinal dimension of the research design	3.3.4
Describes interactions between researcher(s) and host organization(s)	3.3.1
Explains research cycles or phases, if any, and their relationship to the intervention(s)	3.3.5
Presents a clear chain of evidence from observations to findings	3.3.5
Reports participant or stakeholder reactions to interventions	3.3.3

Reports lessons learned by the organization	3.3.5
Researchers reflect on their own possible biases	3.3.1
Provides supplemental materials such as interview guide(s), coding schemes, coding examples, decision rules, or extended chain-of-evidence tables	3.3.4
Uses direct quotations extensively	N/A - direct quotations can only be obtained once the empirical evaluation has been completed.
Validates results using member checking, dialogical interviewing, feedback from non-participant practitioners or research audits of coding by advisors or other researchers	3.3.5
Findings plausibly transferable to other contexts	3.3.5
Triangulation across quantitative and qualitative data	3.3.5

The Empirical Standards Framework’s Design Science Standard ¹, described in Chapter 1, requires that a technological solution be empirically evaluated. The Interruption Management System (IMS) prototype has been developed using the Design Science Standard, so it must be empirically evaluated. The Action Research Standard, which is also defined by ACM SIGSoft, can be used to empirically evaluate

¹Available on <https://github.com/acmsigsoft/EmpiricalStandards>

a technological solution, which is the IMS prototype. Each attribute of the Action Research Standard is listed in Table 3.2 with the corresponding section number of a proposed empirical evaluation plan. The main standards, described by the attributes, are researcher metadata, site selection, interventions in the evaluation, capturing findings during the evaluation, and evaluating the findings from the evaluation will be discussed. The empirical evaluation plan that is defined in this thesis can be used in future studies to determine if the tool is usable for mentors in a real-life context. The evaluation of this prototype is not in the scope of this thesis, so a proposed evaluation will be discussed instead. Ultimately, this evaluation design can be used to empirically evaluate whether the IMS prototype will be useful for mentors in the STaBL Foundation.

3.3.1 Researcher Metadata

Researcher Profile

The researcher(s) that will be conducting the experiment will be from Dr. Anand's lab at McMaster University. Typically members of Dr. Anand's lab vary in academic experience, and may be in the process of completing an undergraduate or graduate degree. The members of the lab are also typically associated with STaBL Foundation and will participate in facilitating Elm Workshops or Learning Sessions in (virtual or in-person) classrooms. The researcher(s) for this study will interact with STaBL Foundation through in-person dialogue, virtual calls, and through electronic messaging tools such as Microsoft Teams or Mattermost.

Researcher Biases

Researcher(s) that have previous mentorship experience may be prone to observer bias as they may expect a particular response or action when they are observing the mentor participants in the study. Researcher bias may also enter the question-designing aspect of the study if researchers use past experiences to influence the design of questions. Researcher(s) can help mitigate bias by practicing triangulation across methods through multiple trials, multiple researchers conducting interviews, and collecting data from multiple experiments. If researcher(s) are noticing different results after conducting an interview, then it is important to further reflect on the results to determine why they are so different. Additionally, a semi-structured interview may help capture deeper findings about why a mentor had a specific experience. There may be situations where an impactful experience in the classroom, aside from the usage of the tool, resulted in a particular association with the tool.

3.3.2 Site Selection

Elm Workshops

The site that will be selected for the evaluation of the Interruption Management System (IMS) prototype will be one or more “Elm Workshops”, which are instructional sessions where mentors guide students through a predefined lesson plan. STaBL Foundation frequently visits classrooms, virtually or in-person, to administer “Elm Workshops”. Examples of lesson plans include developing comics, adventure games, and a *Lunar Lander Activity* where students will design and implement two “views” of a lunar lander (top and side), and write a function to capture the lunar lander’s movement across a digitally pre-rendered moon.

To evaluate the IMS prototype, there will be several “Elm Workshops” organized which will consist of mentors guiding students, in a virtual and in-person setting, through a lesson plan. The usability of the IMS prototype may differ between a virtual and in-person setting, so it is necessary to have “Elm Workshops” in both settings.

As of May 2025, mentors in the STaBL Foundation conduct “Elm Workshops” in two virtual classrooms, once a week, and each virtual classroom consists of Grades 4 to 8 students. The two virtual classrooms can be used to evaluate the IMS prototype; mentors can use the IMS prototype to make announcements and collect feedback from the students in the classroom instead of randomly interrupting the students by speaking aloud in a group call, or messaging the students in a group chat on a video streaming service.

To evaluate the IMS prototype in an in-person, “Elm Workshops” will need to be delivered in an in-person classroom. Mentors will continue to instruct students using a pre-defined lesson plan, which can be the same as the virtual classroom visit, but will use the IMS prototype to collect feedback and make announcements. It is not necessary that the same lesson plan as the virtual classroom visit be used for the in-person classroom visit, however it will make each empirical evaluation more consistent because announcements and feedback interruptions will be delivered at generally the same time to the students.

3.3.3 Interventions

Intervention Purpose & Description

The general purpose of the tool is to interrupt the students participating in a workshop with the least amount of disruption so that mentors can make announcements or collect feedback. As there are multiple users interacting with this tool, mentors, teachers, and students, there needs to be a focus on evaluating the experiences of each user. The scope of this study is to evaluate whether the mentors can effectively use the tool to schedule announcements and to collect feedback from students. The evaluation of the quality of responses in the feedback the students provide is not in the scope of this study, because the students' responses are a reflection of the quality of questions that are asked.

As such, this study's intervention scope is to evaluate the experiences of mentors and students, through semi-structured interviews (further discussed in **Intervention Evaluation**), to determine its effectiveness and usability.

Mentors will use the tool in the "Elm Workshops" to periodically send announcements and to collect feedback from students. For their convenience, a pre-defined list of questions is made available to the mentors in Appendix A that can be used to collect feedback, however mentors can also ask any question they want to the students using the tool.

Students will indirectly use the tool, while they are being interrupted, and mentors are expected to communicate the students' experiences through the aforementioned semi-structured interviews. Justification to not ask students to participate in the semi-structured interviews is listed and discussed in **Intervention Evaluation**.

Intervention Evaluation

Mentors will be asked to participate in a semi-structured interview to determine the effectiveness of the tool in the real-life context of the “Elm Workshop”. Similarly to the study design described in Section 3.2, thematic analysis will be applied to the transcripts from the semi-structured interviews. The thematic analysis will inform whether the Interruption Management System (IMS) prototype is effective or not, and will inform the next design iteration. Questions for the semi-structured interview will be sourced from the statements in the Design Thinking Question Bank, available in Appendix A.

Students will not be asked to participate in the semi-structured interview because it is expected that the mentors will provide enough feedback about the holistic experience of the classroom in the semi-structured interviews. Mentors learn about the students’ overall experience during the workshops because they are constantly engaging with the students to resolve programming errors. Anecdotal sentiments about a student’s workshop content and experience can be collected at this time, and subsequently communicated during the semi-structured interview. Interviewing students, particularly those in K-12, may introduce ethical and logistical complexities with consent and data quality. Mentor observations provide a more ethically viable and contextually informed alternative.

3.3.4 Capturing Findings

Longitudinal Dimension

The same mentors will be participating in all the “Elm Workshops” in this study, and the construction of the semi-structured interview questions will need to take into

consideration the necessary approach to collect feedback from the mentors. Mentors can participate in a semi-structured interview after each workshop session, where it will be necessary to design different questions for each workshop, or mentors can be asked questions after all the workshops have been completed. If mentors participate in an interview right after a workshop, then they will remember much of the session and be able to give more detailed answers. If mentors participate in an interview at the end of all the workshops, they will be able to provide more holistic answers. In order to capture the experience of the mentors throughout all the workshops, it is recommended to capture their experiences after each workshop to determine if there are any repeating pain points. If time permits, conducting an interview at the end of a workshop will allow the mentor to further illustrate the severity of each pain point.

Supplemental Materials

The researcher(s) conducting this study can source questions for the semi-structured interview from the Design Thinking Question Bank in Appendix A.

3.3.5 Evaluating Findings

Research Cycles & Phases

A new set of features implemented into the Interruption Management System (IMS) prototype signifies the start of a research cycle. The new set of implemented features must be empirically evaluated in a, virtual or in-person, classroom during an “Elm Workshop”. It is difficult to determine *exactly* how many workshops must be completed before conducting semi-structured interviews with mentors to inform the next design iteration of the IMS prototype. If a new set of introduced features is strongly

misaligned with the users, then even one workshop will be enough to understand that there is a strong aversion to the new functionality. However, if there is no strong aversion from the users to a new set of features, then it may take more than one workshop to identify a new set of features for the IMS prototype that are reflective of the users' pain points. The process to develop the next design iteration of the IMS prototype is more of an art than a science. The steps to complete one research cycle are outlined as:

1. Mentors instruct lesson plans in “Elm Workshops” with the IMS prototype.
2. Mentors are invited to participate in short semi-structured interviews.
3. Apply thematic analysis on interview transcripts.
4. Repeat steps #1 to #3 until patterns are identified using thematic analysis.
5. Map themes from thematic analysis to new IMS prototype features.
6. Implement new IMS prototype features.

Reporting Lessons Learned, Observations, & Evidence

Transcripts of the semi-structured interviews with mentors will be the primary source of evidence for the subsequent thematic analysis in each research cycle. Transcripts are critical for thematic analysis, and can also contain sensitive information such as names, so they must be stored in a secure digital storage solution for future use. Common workflow tracking tools such as JIRA, MatterMost, or GitHub will be used to track each feature request for the Interruption Management System (IMS) prototype. Each of the aforementioned tools has the capability to track a theme identified during

the thematic analysis phase to a new feature request. In alignment with the Design Science Standard, the new feature should follow the format: “As a <USER>I want <FEATURE>so that <VALUE>”. In this format, the “USER” is the primary user that will be benefited from the feature, “FEATURE” is a description of the feature that needs to be implemented, and “VALUE” describes benefit the feature will provide to the user. The themes, which are the outcome of the thematic analysis, should be used to describe the “VALUE”. Using the aforementioned format ensures that observations and lessons learned from each research cycle are consistently documented, and used to inform the next set of features in the IMS prototype.

Validating Results

Patton [40] has outlined “Triangulation” as using multiple sources of data, theories, methods, and/or researchers/investigators to address a research objective or question. More practically, it is a research strategy which can be used to mitigate bias and improve the credibility of any findings.

Denzin [14] and Patton [40] indicated that there are four types of Triangulation which can be applied to qualitative research: (a) multiple sources of data, (b) multiple theories, (c) multiple methods, (d) multiple researchers/investigators. Triangulation can be done in one of the four aforementioned types or can happen across multiple types. That is, to bolster the validity of findings data source triangulation and methodological triangulation can be used instead of just one. However, it is common for researchers to use just one type of Triangulation as it can be complicated to ensure consistency in the design of studies when using multiple triangulation methods, and it can lengthen the time of the research study [4]. Researchers in the study will use one

of the methods of triangulation based on how many members they have participating in the Elm Workshops and how many Elm Workshops are conducted.

The easiest ways to employ triangulation are through data source triangulation and researcher/investigator triangulation.

Data source triangulation involves collecting data from multiple sources, which may be different participants, locations, and/or times, to examine the same phenomenon [8]. Findings can be more reliably validated by ensuring that results are consistent across various groups or contexts. In the context of this study, semi-structured interviews with multiple mentors will be conducted to improve the validity of the results. Data source triangulation helps to confirm whether similar themes and topics emerge across interviews.

Research/investigator triangulation refers to involving multiple researchers in the study to reduce individual researcher bias [8]. For example, if multiple researchers code the same interview transcripts and arrive at the same conclusion, then the findings are more trustworthy. In this study, multiple researchers should conduct research cycles, described in Section 3.3.5, and there should be more than one researcher involved in each research cycle step to reduce researcher bias.

Chapter 4

Software Technology Stack

In this chapter, the functional “building blocks” for the Interruption Management System (IMS) prototype, and where it has been integrated into applications built by the STaBL Foundation are listed and discussed. **Elm & Model-View-Update** discusses the Elm functional programming language and the Model-View-Update (MVU) paradigm that was used to develop the IMS prototype. The MVU paradigm is used to render updates to the user interface (UI) components of the IMS prototype when there is a change in state. **GraphicSVG** is a UI components library maintained by the STaBL Foundation, and is imported in the IMS prototype to build more complex UI components. **TEASync** is a framework that is used by the IMS prototype to establish bi-directional communication between users, and is built on-top of The Elm Architecture (TEA). **State-Diagram Draw** (SDDraw) is an existing application built by the STaBL Foundation, accessed through STaBL.rocks, and the IMS prototype is integrated into SDDraw to schedule and deliver interruptions to users of the application.

4.1 Elm & Model-View-Update

Model-View-Update (MVU) is a Graphical User Interface (GUI) paradigm, which was introduced in the release of the Elm functional programming language, is also known as The Elm Architecture (TEA). In any Elm program, the logic can be informally broken down into three distinct components: (a) *Model* which is the application state, (b) *View* which is a method to turn the state into HTML, and (c) *Update* which is a way to update the state based on a message. This three-step approach is further illustrated in Figure 4.1, which illustrates how a message update to a Model, results in an updated Model with a subsequently updated view. More specifically, the application state in Elm is entirely represented within a Model and a new Model is created as a result from an Update function. The Update function takes a Model and a Message as an input and outputs a new Model which consists of the necessary transformations. Messages are used to communicate events from a web application, such as button clicks or keyboard input, and provide Model-Update instructions to the Update function. A View is generated from each Model and is presented to the user, which takes a Model as an input and generates HTML that is used to render the web application. The Model is immutable, which helps minimize any side effects that result from a program written in Elm. This feature aligns with the principle of being a Pure functional programming language.

4.2 GraphicSVG

GraphicSVG is a library based on the original Elm Graphic module and is developed and maintained by the STaBL Foundation [51, 12]. GraphicSVG uses two main types

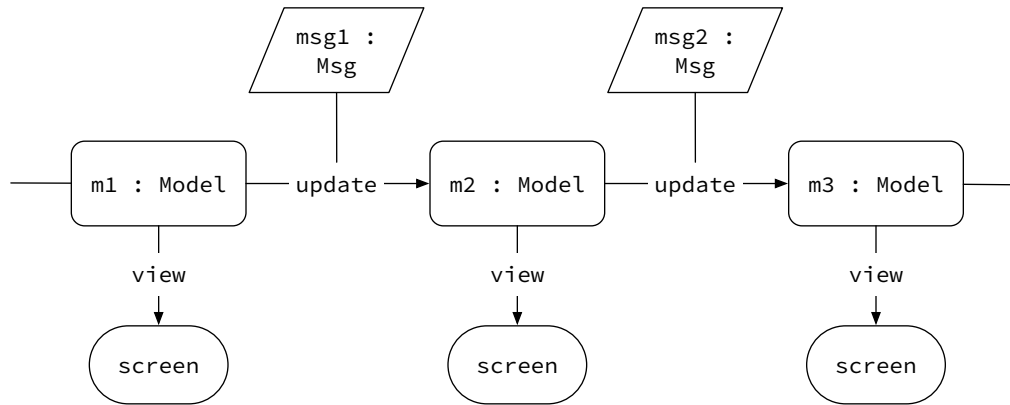


Figure 4.1: Model-View-Update

to match students' (who are the primary users) intuition about visual arts: the *Stencil* type and the *Shape* type.

Stencils capture structural properties about the Shape that a user would want to create by accepting structural information as inputs. For example, the **Circle** and **Square Stencils** will take a **Float** as an input, representing the radius and side length respectively. Whereas a **Rectangle Stencil** will take in two Float input parameters, representing the width and length. Stencils do not contain any properties about the colour or transformations to the Shape, and any required colours or transformations must be applied to the Stencil. The code below is an example of a circle of radius 5, filled with the colour blue, moved to (3,5) coordinates, and scaled to 0.5 of it's original size:

```
circle 5
  |> filled blue
  |> move (3,5)
  |> scale 0.5
```

The result of the above Shape is displayed in Figure 4.2 inside of the STaBL.rocks IDE.

Ultimately, GraphicSVG is a toolbox of functions that can be used to develop complex applications inside of the STaBL.rocks IDE. The library has been updated several times since the release of the original paper “Graphics Programming in Elm Develops Math Knowledge & Social Cohesion” [51], but this paper still outlines its core capabilities.

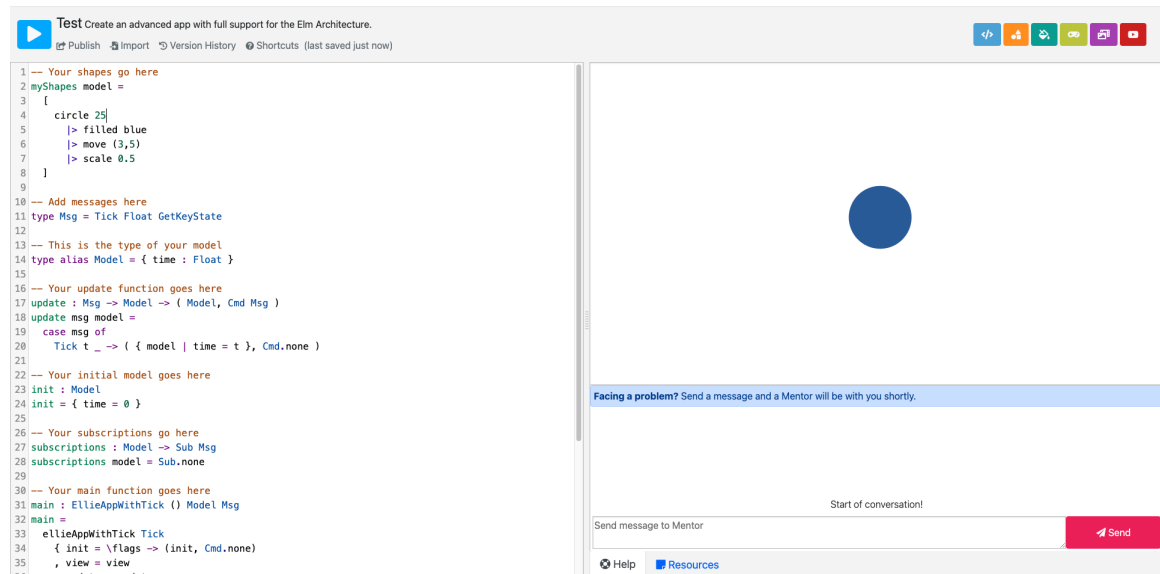


Figure 4.2: Example of the STaBL.rocks IDE which can be used to render GraphicSVG Shapes.

4.3 TEASync

Schankula et al. [46] developed TEASync, which is a framework for multi-user games

that is built using GraphicSVG and The Elm Architecture (TEA). While TEA is a Model-View-Update (MVU) framework, TEASync leverages a new classification titled *Local-Global Model-View-Update* (LG-MVU) which is a new design pattern for distributed and multi-user applications. In a simple application, each client is running its own instance of the application without being exposed to details of the other client's application instance. For example, if Client A creates a blue circle on their screen, then naturally Client B will not be aware of those details as their client systems are not connected in anyway. That is, each Client will have a local model that they only have access to, but no global model, which contains information about each other's applications. TEASync enables client-to-client communication by maintaining a global model where each Client is able to update and read state. State updates result in changes being made available on both Clients' systems so that there is bidirectional communication. TEASync also supports an Integrated-Development-Environment (IDE) that enables safe synchronization by employing type-safe encoders and decoders so that the TEASync runtime can maintain a single global state.

4.4 State-Diagram Draw

Pasupathi [38] developed State-Diagram Draw (SDDraw), which is a state diagram editor design for children, using GraphicSVG. The STaBL Foundation later integrated it with TEASync on STaBL.rocks. GraphicSVG and TEASync are described in Section 4.2 and Section 4.3 respectively. Students can use SDDraw to learn about and create state diagrams. The Interruption Management System (IMS) prototype was integrated into SDDraw so that mentors could interrupt students at opportune moments during Elm Workshops, which are described in Chapter 3. The integration

of the IMS Prototype into SDDraw was motivated by the need to evaluate the IMS prototype in a real-life context. While this thesis does not evaluate the IMS Prototype in SDDraw, integration is necessary to facilitate future studies and a proposed plan is presented in Chapter 3 to evaluate the IMS Prototype integration in SDDraw.

4.4.1 Integration with State-Diagram Draw

Since SDDraw is developed and hosted on STaBL.rocks, a module was created for the IMS prototype on STaBL.rocks and imported into the SDDraw entrypoint file for most components. The entrypoint file imports software packages that can be used to develop UI components that are rendered when a student accesses SDDraw, and is titled “Main.elm”. The “Main.elm” centralizes the UI components for the integration of the IMS prototype into SDDraw.

4.4.2 Technical Limitations & Workarounds

The components that were unable to be imported into SDDraw from the IMS Prototype were the Local Model Types and Global Model Types, and their corresponding update functions. A limitation of STaBL.rocks is that Types cannot be imported from modules, so they were copied and pasted into the SDDraw Types file, which was necessary to resolve compilation errors. Another limitation was the unavailability of the Pull type alias in the GraphicSVG module via TEASync, which prevented the integration of the bounded deferral emoji used in Prototype #3 (Chapter 9). The new bounded deferral emoji only contains aesthetic changes so there is no impact to existing functionality, however it may be less appealing to students, as pointed out by a mentor in Chapter 8. Otherwise, the integration was straightforward because

minimal changes were required to integrate the remaining components of the IMS prototype into SDDraw.

Responses from students were not captured in a permanent storage solution, such as a database, because it is out of the scope of the IMS prototype since the responses are being stored in the Global Model. To permanently store responses from students, the TEASync framework must store the Global Model in a permanent storage solution.

4.4.3 Hierarchical Task Analysis & Interruption Delivery

Hierarchical Task Analysis (HTA) was applied to SDDraw to determine an opportune moment to interrupt a student before the bounded deferral timer expires. The Elm Architecture (TEA), discussed in Section 4.1, displays a UI screen based on the current state of the application. In SDDraw, there are only two states where the student is not engaged in creating a state diagram, and that is when they are in the “Waiting” or “HelpInstructions” states. SDDraw has the “Waiting” state when they are not actively creating, dragging, or adding transitions to new or existing state nodes. SDDraw transitions to the “HelpInstructions” state when a student clicks on a button to open an “Instructions” screen that explains how to use SDDraw. The IMS prototype will deliver a notification to the student, if an interruption exists in the Global Model interruption queue when SDDraw is in the “Waiting” state or the “HelpInstructions” state, and the bounded deferral timer has not expired yet. If the bounded deferral timer expires, then a notification will be delivered to the student irrespective of the state SDDraw is in. A point of confusion is the usage of the term “state” because state is used to refer to the Model states in TEA, and also a state

which a user creates in SDDraw. States created by the user in SDDraw will be referred to as “SDDraw States” to prevent confusion.

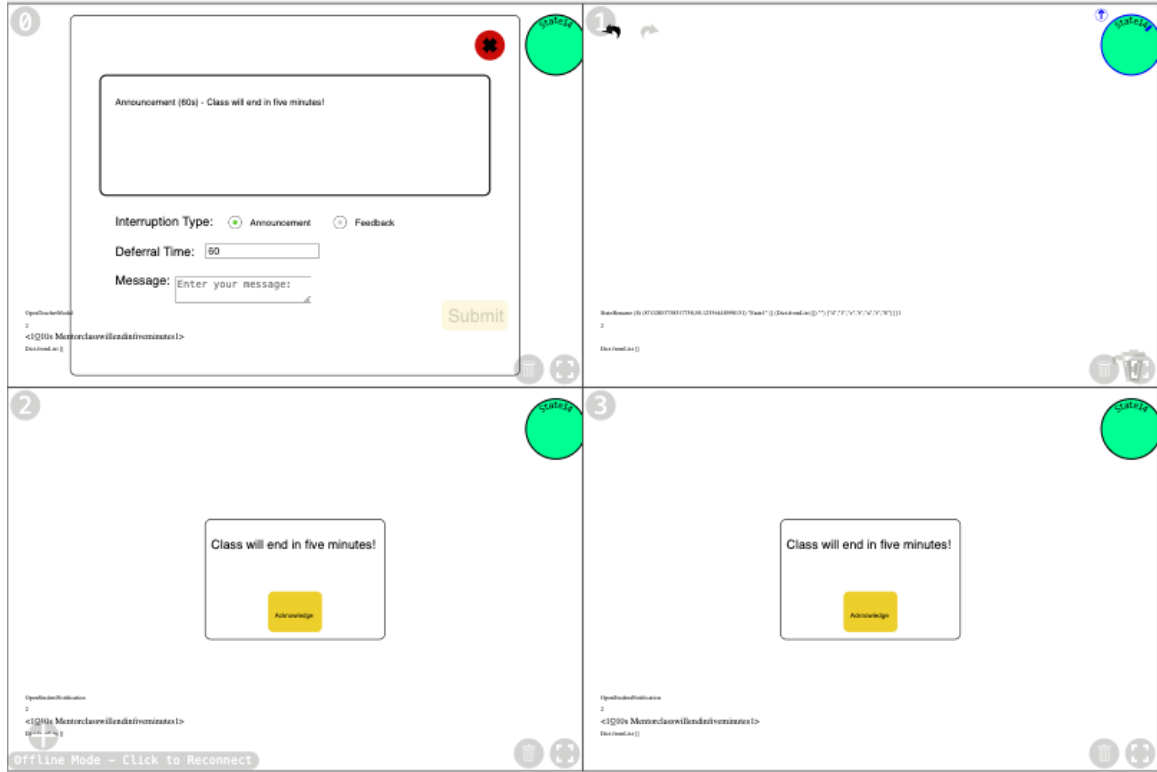


Figure 4.3: Screenshot of four users using State-Diagram Draw with the Interruption Management System prototype integrated. The top-left user is a mentor that has just submitted an interruption. The top-right user is a student that is in a drawing state and is not interrupted; they will be interrupted once the bounded deferral timer expires. The bottom-left user is a student that is in the “Waiting” state and is interrupted immediately. The bottom-right user is a student that was reading the instructions in the “HelpInstructions” state and is interrupted immediately.

Figure 4.4 is a screenshot of SDDraw in the “Waiting” state with the button to transition to the “HelpInstructions” state highlighted with a red-outlined circle. Figure 4.5 is a screenshot of SDDraw in a state other than “Waiting” or “HelpInstructions” where the user is creating a SDDraw State; students will not be interrupted in



Figure 4.4: State-Diagram Draw (SDDraw) “Waiting” state with button to transition to the “HelpInstructions” state highlighted using red-outlined circle. Users will be interrupted in either of these two states when an interruption is present in the Global Model interruption queue.

this state unless the bounded deferral timer for an interruption expires. Figure 4.3 is a screenshot of four users, three students and one mentor, using the SDDraw with the IMS prototype integrated.

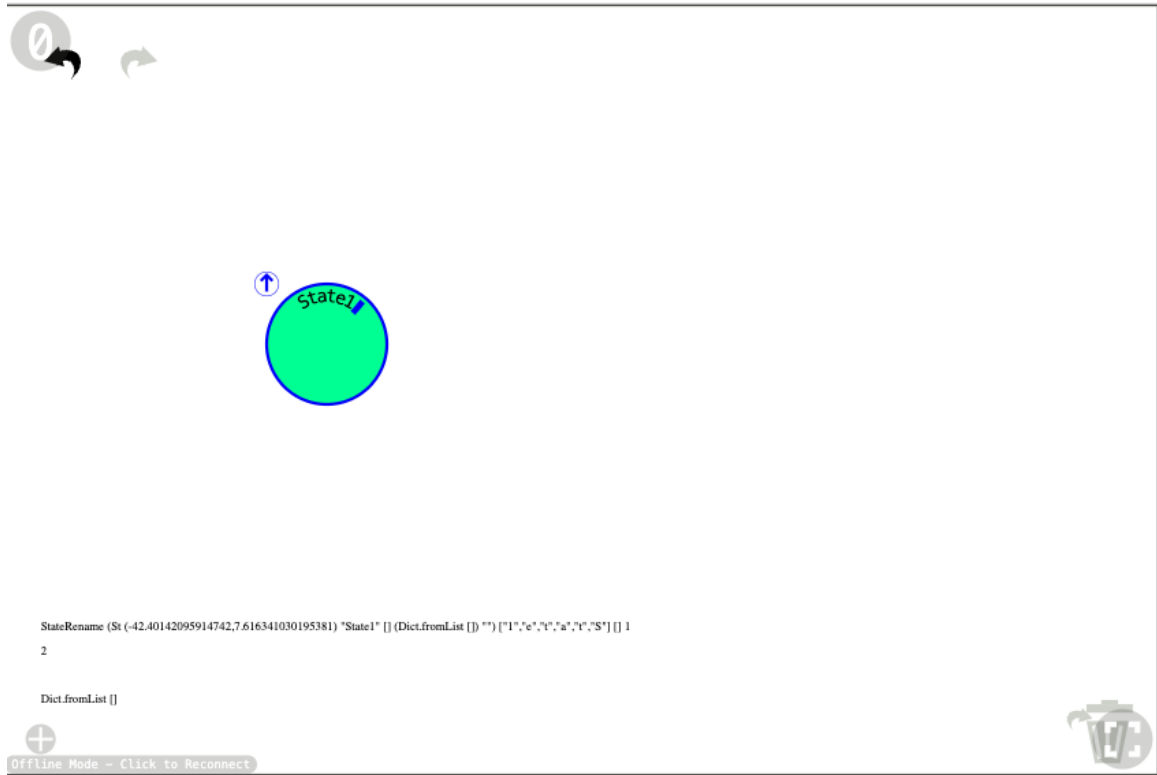


Figure 4.5: State-Diagram Draw (SDDraw) in a state other than “Waiting” or “HelpInstructions”, where the user is creating a SDDraw State. A user will not be interrupted in this state unless the bounded deferral timer for an interruption expires.

4.5 Teacher Dashboard & MultiShapeCreator

Although the Interruption Management System (IMS) prototype was not integrated into the Teacher Dashboard or MultiShapeCreator (MSC) due to the significant development rework required, this section outlines how the two systems work and how the IMS prototype might be integrated into the two for future studies.

The Teacher Dashboard is an application of TEASync in the context of observing the work that students are completing within a classroom. Previously, teachers or

admins didn't have the ability to view what students were doing unless they physically walked over to their computer. During COVID-19, physically viewing what the student was doing was not possible since students were completing work remotely and sometimes asynchronously. The Teacher Dashboard allows a user with elevated permissions to view the content of another user while they are using the IDE to determine whether they are working towards completing their task or not. Teachers (who have the admin role in the STaBL.rocks IDE) will log in to the Teacher Dashboard (Figures 4.6 and 4.7) and initialize the environment. Teachers can then add students to the multi-user environment by following the natural prompts within the dashboard (Figure 4.8). Figure 4.9 shows four users or students that have been added to the Teacher's Dashboard. Once the students have been added, they can then navigate through multiple options to choose what they would like to do (Figure 4.10). Students will have the option to use the MSC, which refers to the MultiShapeCreator, change their avatar, create animations or pictures from a Wordathon, or look at their Classroom (Figure 4.11).

MultiShapeCreator (MSC) is a tool built by the STaBL Foundation using GraphicSVG, which is described in Section 4.2, to help students create GraphicSVG Shapes and animations. Shapes can quickly become complicated to build and can deter students from attempting to build more complicated animations or pictures. MSC originally began as the Shape Creator, which allows students to build shapes for their individual programs without any supervision. MSC leverages TEASync to provide an interactive way for a teacher to view what their students are doing via the Teacher Dashboard. The following sections describe how a user is expected to interact with the MSC when using it on STaBL.rocks. **Landing Page** describes the UI page that

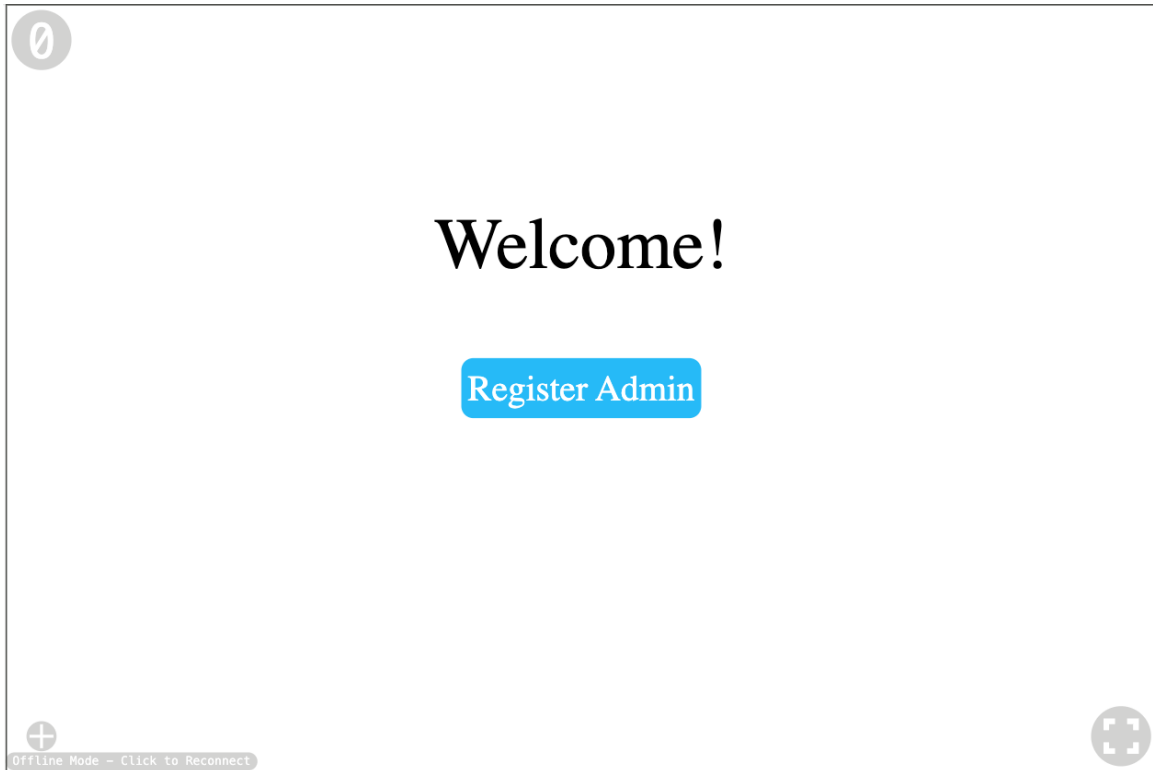


Figure 4.6: Teacher Dashboard landing page.

users will first see when they “land on” or open MSC. **Default Page** describes the next screen the users will interact with, after they have navigated through the landing page. **Creating Shapes** describes how a user will use the MSC to create a GraphicSVG Shape and integrate it into their existing Elm program.

4.5.1 Landing Page

Figure 4.13 contains the landing page view of what users will see when they first load MSC; users will see instructions on how to use MSC. Specifically, the instructions are to click, tap or use arrow keys and return/enter to choose:

1. A Stencil

0

Set teacher login.

TeacherDefault

teacherdefault

Register

Offline Mode - Click to Reconnect

Figure 4.7: Teacher Dashboard login page where the username and password are set to TeacherDefault and teacherdefault respectively.

2. Text transformations if you choose the Text stencil
3. To fill or to outline
4. With any colour
5. Geometric transformations
6. Copy and paste the code into your program

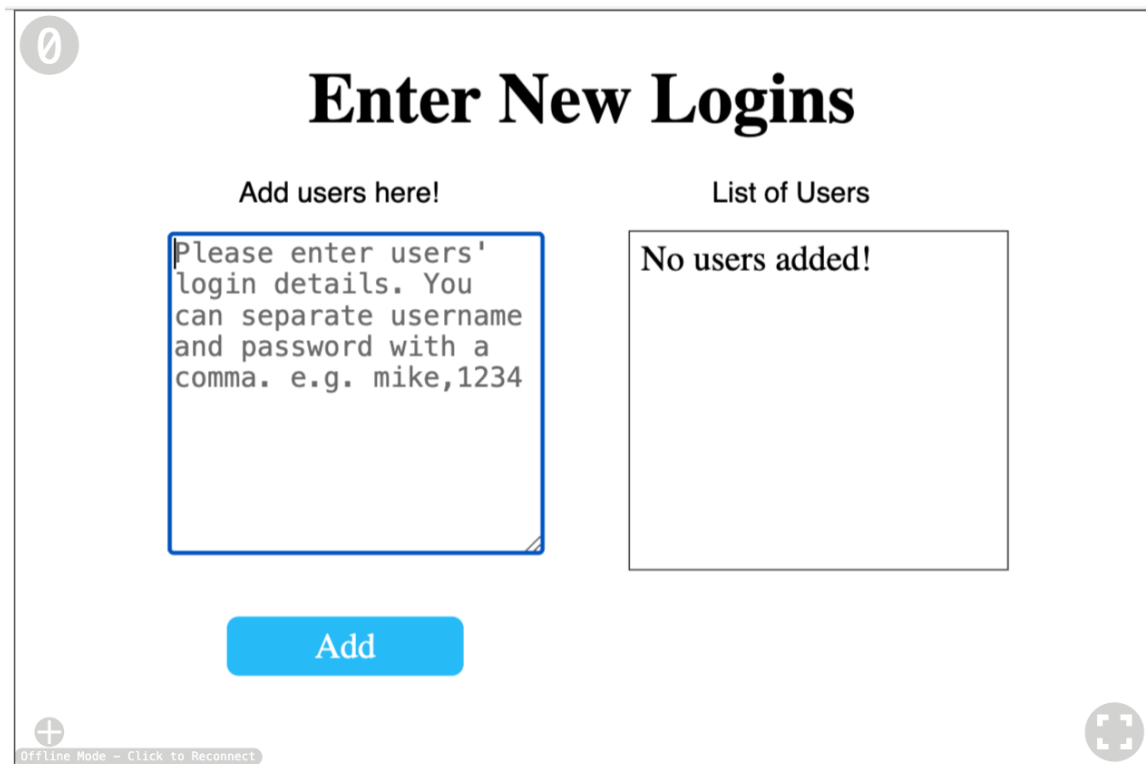


Figure 4.8: Teacher Dashboard page where new users can be added.

4.5.2 Default Page

Once a user has reviewed the instructions on the landing page, they can click anywhere on the page to view the default page which is captured in Figure 4.14. The default page is organized logically so that users can naturally follow the instructions on the landing page, without having to constantly reference it. The default Shape that has been created is of a red outlined circle with a radius of 10 units. The *Wider* and *Narrower* buttons can be clicked to increase or decrease the radius of the circle. The code that a user can copy has been also generated in the bottom left of the screen and is as follows:

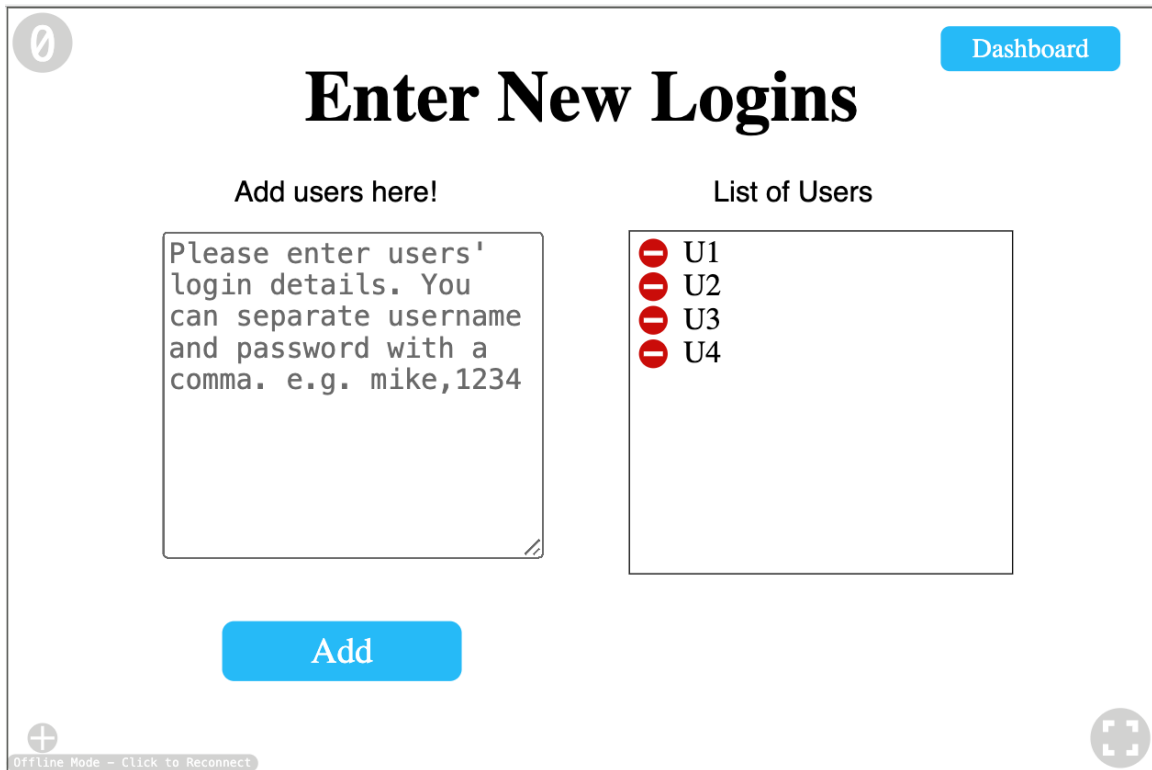


Figure 4.9: Teacher Dashboard web page where four new users have been added; U1, U2, U3, and U4.

```
[ circle 10
  |> outlined (solid 1)
    red
]
```

4.5.3 Creating Shapes

Aside from the default settings, users have the option to create many different kinds of shapes and apply many different types of transformations to each Shape. The Shapes can become quite complicated, see Figure 4.16, where Shapes that would be

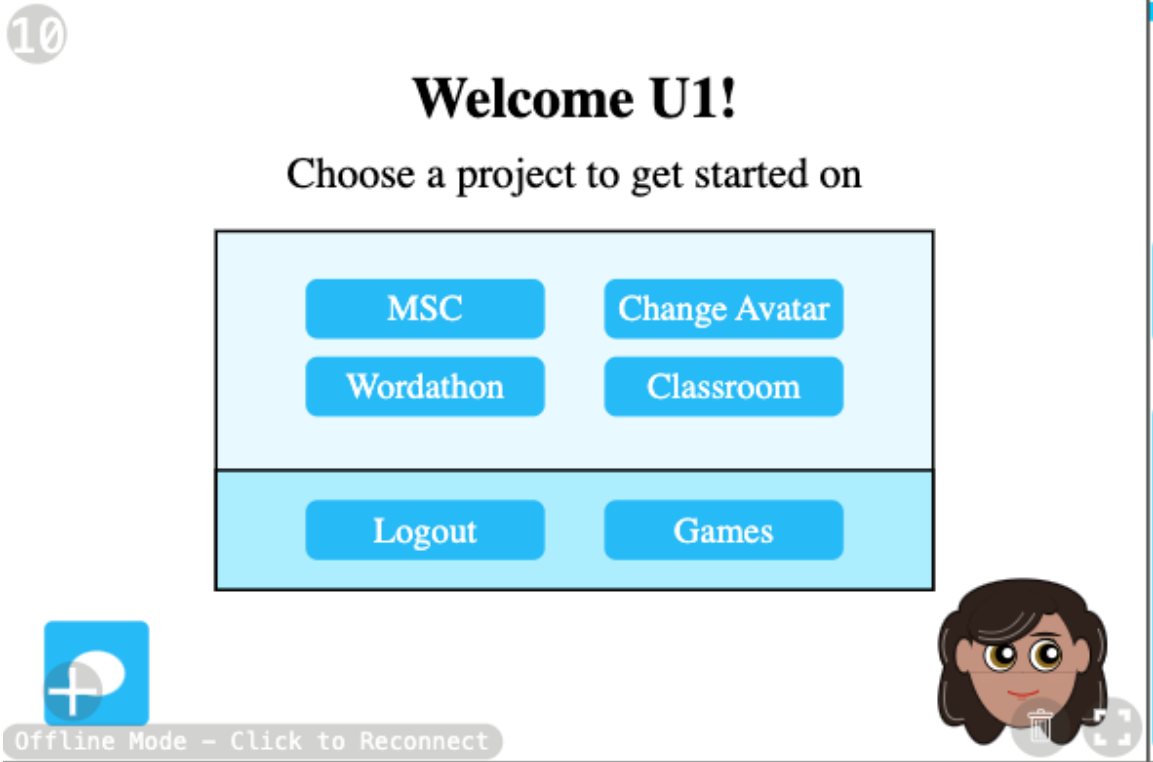


Figure 4.10: The Welcome page that a new user or student will see when they first sign into STaBL.rocks and access the link that the admin or teacher provides them.

difficult to normally create with just code can be visualized and interactively created using MSC. Students can copy the code for very complicated Shapes directly into the software that they are writing.

4.5.4 Exploiting Task Phases in MultiShapeCreator

This section discusses how task phases can be exploited in the MultiShapeCreator (MSC) by identifying a primary goal and tasks that are required to complete the respective primary goal. Ultimately, these findings will be mapped back to MSC to determine when an interruption should be delivered.

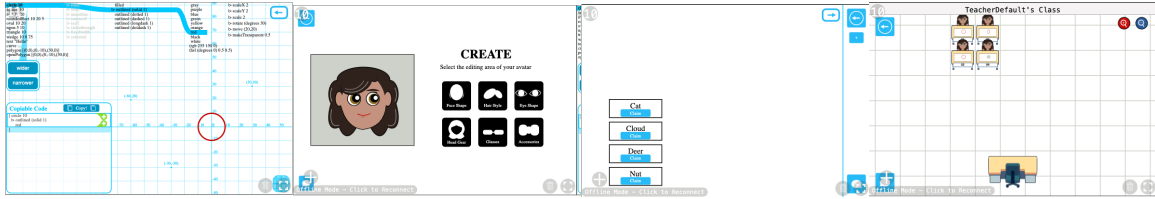


Figure 4.11: The four choices that a user or student currently has in the Teacher Dashboard from left to right are to: (a) Use the MultiShapeCreator, (b) Change their avatar, (c) Create an animation/picture using a word from the Wordathon, and (d) View their classroom

The Seven Stages of Action developed by Norman and Miyata can be used to identify when the right time to interrupt a user is, by identifying which stage or task phase they're in. Goals and tasks can be identified by using Hierarchical Task Analysis (HTA), which is initialized by identifying a goal that the user must complete in a specific process. There are subsequently tasks and sub-tasks defined that the user must complete in order to achieve their goal. The least amount of disruption to a user typically is when a user is in the process of forming a new goal or sub-goal. Another heuristic to identify when a user will be less disrupted is to understand where there is a lower amount of cognitive load on the user. There is a detailed review of the Seven Stages of Action and HTA in the Literature Review section.

A student's primary goal in MSC is to create a shape which can be pasted into the software program that they are building. There are five tasks that are identified which need to be completed with an assortment of sub-tasks that must be completed for each task. The details of the individual tasks that a user needs to complete in order to create a shape are identified in the Figure 4.15. These tasks were identified by identifying where a student would need to click in order to move onto the next step of the MSC. For example, in order to select a Shape stencil, a student must click on the stencil that they would like to use and then to click on the respective buttons to

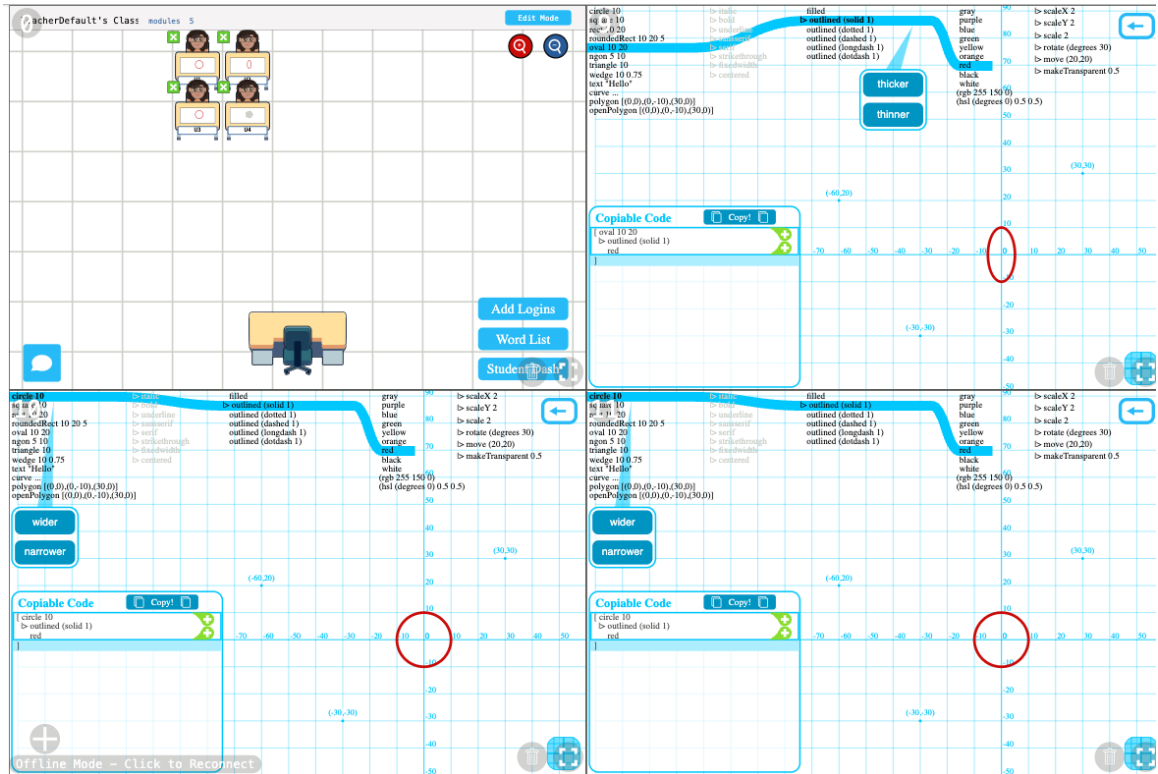


Figure 4.12: Example of the Teacher Dashboard with four students that are using the MultiShapeCreator.

adjust the dimensions of the shape stencil until their desired dimensions are reached. If the user wants to use the default dimensions then they do not need to engage in the sub-task of adjusting the dimensions. As such, there are at least two clicks required if the user wants to adjust dimensions of their shape stencil and only one click required if they want to use the default dimensions of a particular stencil. While only one goal is identified in Figure 4.15, there are many sub-goals that are implicitly present that the user forms while completing each task; selecting the correct stencil for their software program is an example of such a sub-goal.

Since the least amount of disruption is caused when a user is in between the stages

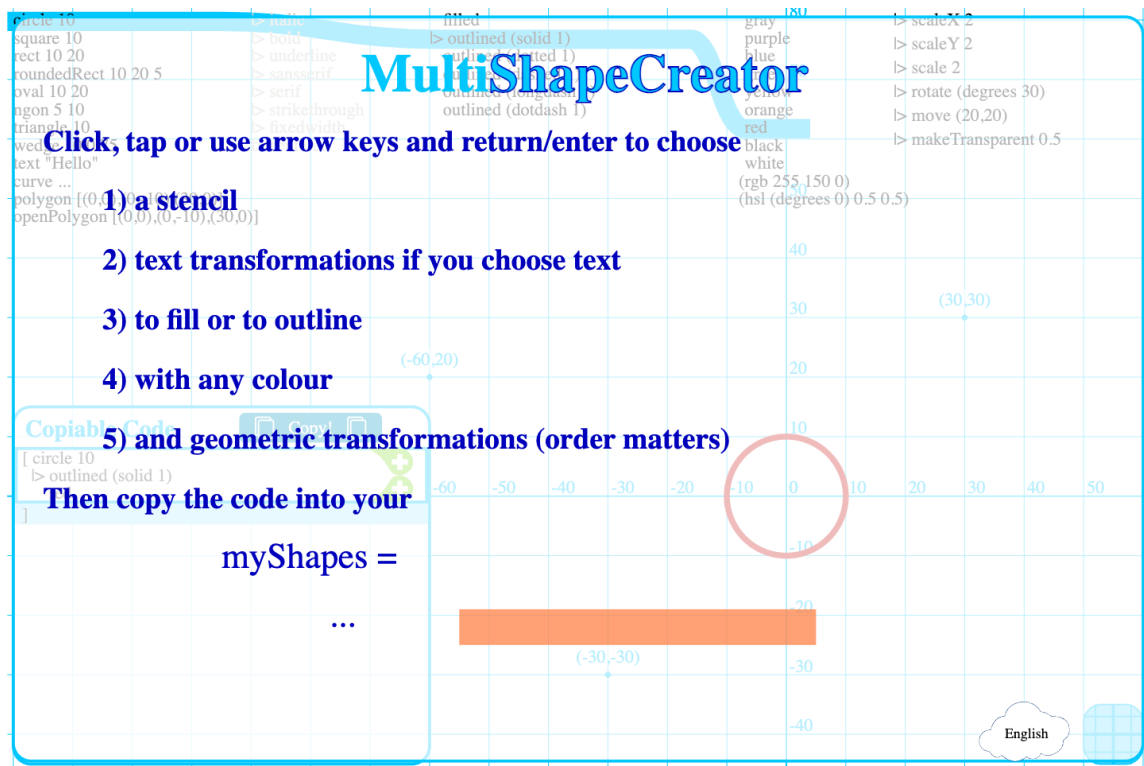


Figure 4.13: Multi Shape Creator (MSC) landing page which instructs students on how to use Multi Shape Creator.

of completing their task and forming a new goal, an interruption should naturally occur after Task #5. Interruptions delivered during these times will naturally result in the least amount of interruption because a user should have the least cognitive load. That is, a user will have completed the task of creating a Shape that they can copy and paste into their software program. At this time, it is theorized that all of the information that they need is externalized inside MSC instead of being in their mind.

Otherwise, less optimal times to deliver an interruption will be when a user is in between task phases. It is also vital to note that it will be difficult to deliver an

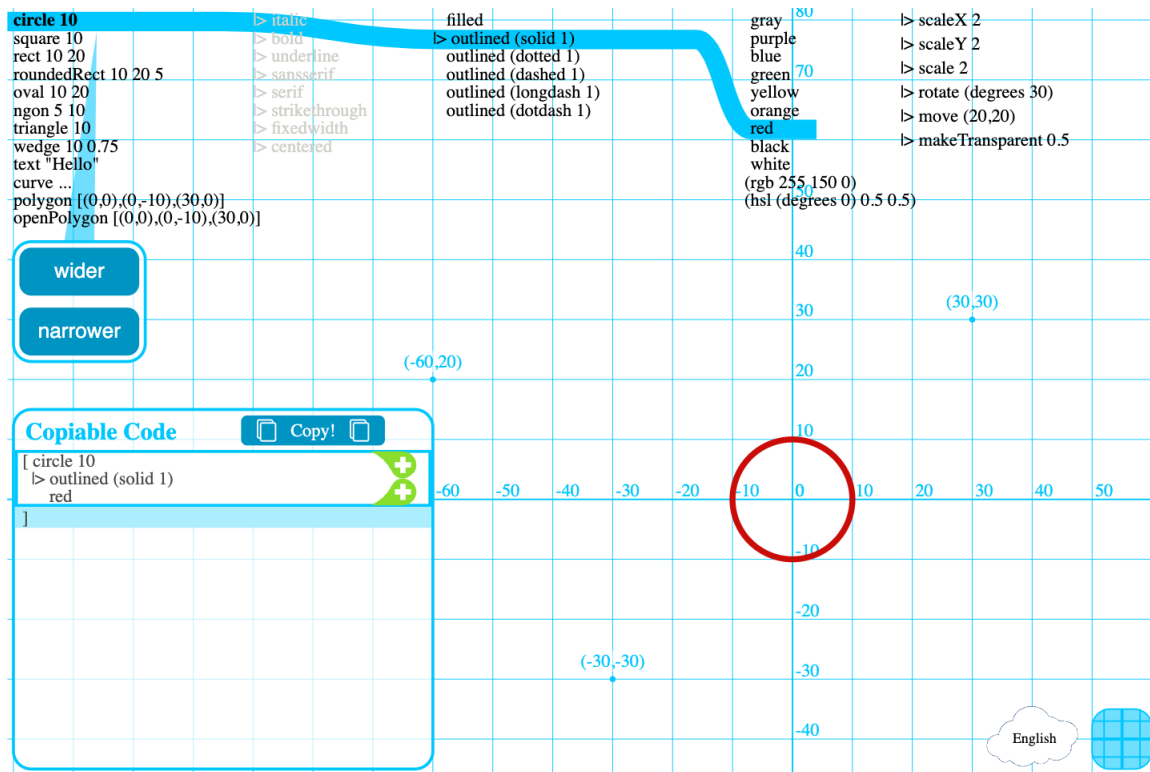


Figure 4.14: Multi Shape Creator (MSC) default page which has a red outlined Circle of radius 10 with no other transformations applied to it.

interruption between sub-task phases because a user will typically click many times to adjust the dimensions or magnitude of a stencil or transformation when they are in MSC. Therefore it will be difficult to understand when that sub-task has been completed, and interruptions during the rapid-click activity will be highly disruptive as a user is in the middle of completing an action.

MultiShapeCreator Goal: Create a Shape

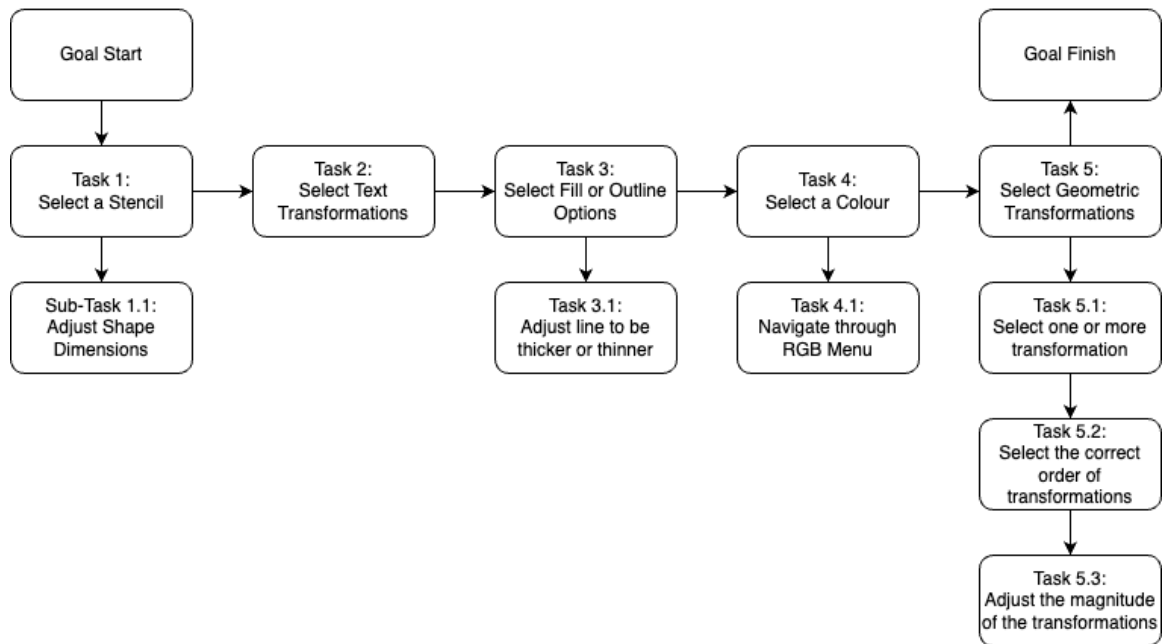


Figure 4.15: Hierarchical Task Analysis (HTA) of the MultiShapeCreator to identify which tasks need to be completed to complete the overarching goal of creating a shape.

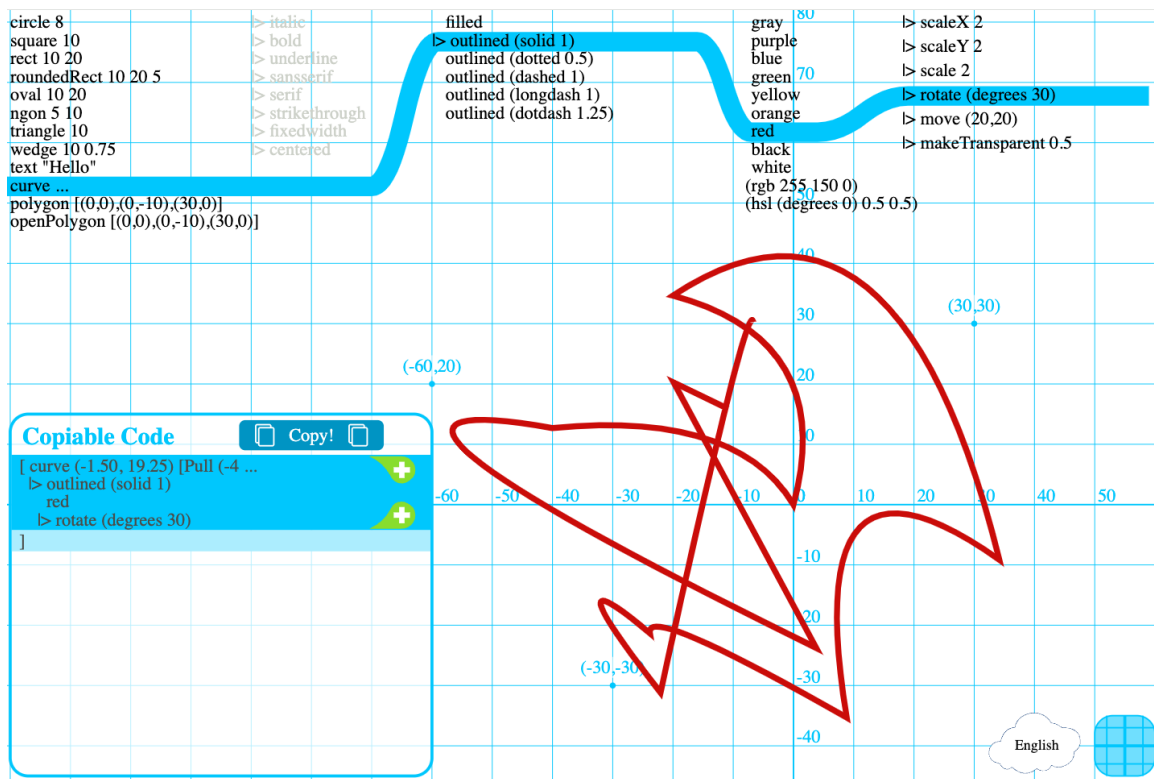


Figure 4.16: MultiShapeCreator Curve Example

Chapter 5

Prototype Iteration #1

The first version of the prototype was built in STaBL.rocks using the Local-Global Model-View-Update (LG-MVU) paradigm. The prototype was implemented as a standalone application to iterate quickly on its design and functionality, before integrating it into other STaBL Foundation applications. The design was inspired from conversations with the Managing Director of STaBL Foundation, and subsequently implemented in the form of two screens: (a) *Teacher Screen* used to set the interruptions, (b) *Student Screen* where the interruptions would be delivered. The prototype is built to meet a list of requirements outlined in Section 3. The first iteration of the Interruption Management System (IMS) prototype was not integrated into the Teacher Dashboard or MultiShapeCreator. Integrating the IMS prototype into the Teacher Dashboard or MultiShapeCreator required significant developer effort, which would slow down collecting feedback from the users for the research cycles. Since the intention of the research cycles was to only collect feedback for the IMS prototype and not the Teacher Dashboard or the MultiShapeCreator, a standalone application was

developed on STaBL.rocks which demonstrates the capabilities of an IMS. A screenshot of the prototype’s homepage is captured in Figure 5.1. On the left hand side there is a button, “Teacher” which will open the Teacher Screen, and subsequently on the right hand side there is a user id form input and a “Student” button which will open the Student Screen.

As discussed in Chapter 2, the three primary components of an Interruption Management System (IMS) are a: (a) Sensor, (b) Model, (c) and Delivery System. In this prototype, the Sensors are the *opportune* features that trigger the Notification Screen when they are clicked, the model is the bounded deferral timer, and the delivery system is the Notification Screen and Student Screen. The Teacher Screen was developed to facilitate usage of the tool in the STaBL.rocks Teacher Dashboard to encourage teachers, mentors, and students to send notifications.

5.1 Teacher Screen

The Teacher Screen is used to create interruptions which can be delivered to students at opportune moments. A screenshot of the Teacher Screen is in Figure 5.2.

There are two types of interruptions available: (a) *Announcement*, which is a read-only interruption delivered to a student, and (b) *Feedback*, which captures input from a student given a specific prompt. Either type of interruption can be selected using a radio button, but the default selection is set to the Announcement type.

The bounded deferral time can also be entered by a user to capture the latest time that an interruption should be delivered by. The default bounded deferral time is set to 60s, so the program will wait for an opportune moment to interrupt the student for 60s, and if it cannot find an opportune moment within that time, then it

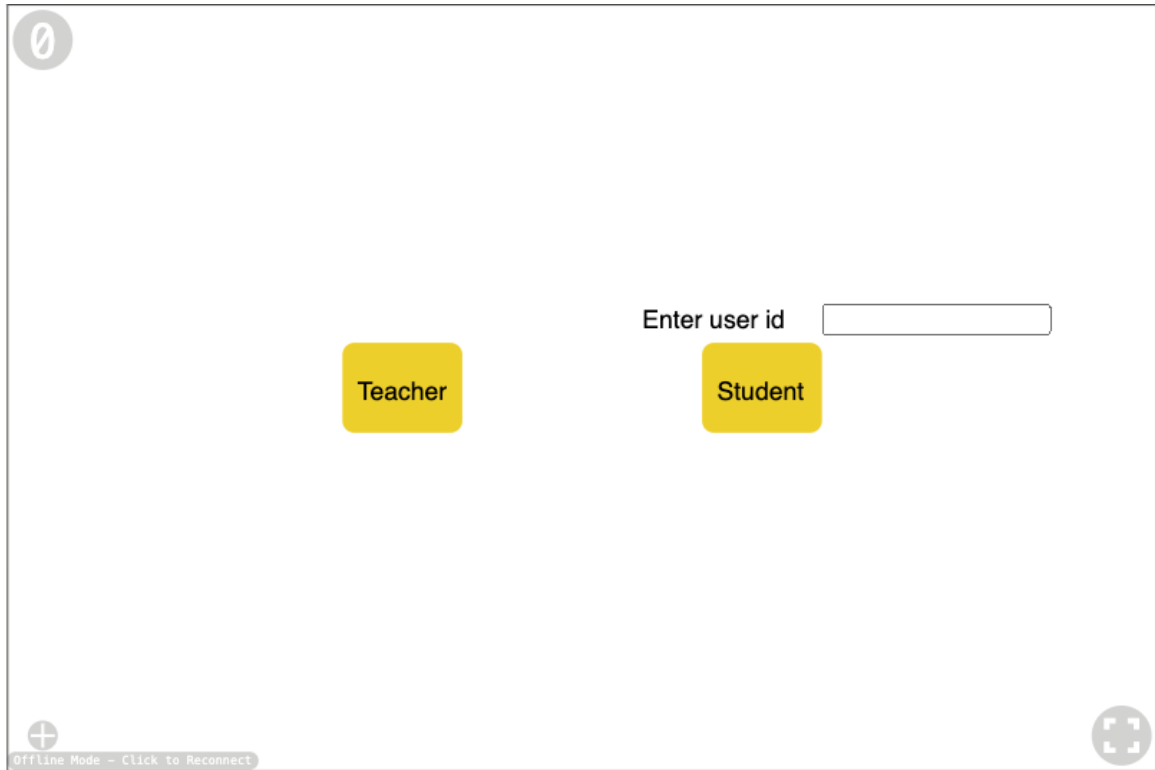


Figure 5.1: The prototype homepage once it has been compiled in STaBL.rocks. On the left hand side there is the Teacher button to trigger the Teacher Screen to open, and similarly there is the Student button the right hand side to trigger the Student screen.

will interrupt the student. Using Hierarchical Task Analysis (HTA), defined in Chapter 2, user interface (UI) components in the MultiShapeCreator (MSC) are marked as “opportunistic” or not. “Opportunistic” UI components are those that a user interacts with when there is a low level of cognitive load, which is the most “opportunistic” time to interrupt a user. When a student clicks on an “opportunistic” UI component, and the bounded deferral timer has not yet expired, the interruption will be delivered to the student. Otherwise, students can always click on the personified emoji to check if they have any new notifications.

Mentors can enter a message at the bottom of the Teacher Screen for the message that they want to deliver to the students. Typically, if the interruption type is an Announcement then the message will be a statement such as “5 minutes left to finish your work!”. If the interruption type is Feedback then the message will typically be a question, prompting a user response, such as “What are your thoughts on the exercise so far?”. An example of what it looks like in the Teacher Screen to enter an interruption is in Figure 5.3; once the user clicks submit, it will add the interruption to the interruption queue.

Mentors can queue up as many interruptions as they want within the Teacher Screen. Once the user creates an interruption then it will be populated in the window at the top of the Teacher Screen, a screenshot of an example is in Figure 5.4.

Mentors cannot delete interruptions and they cannot edit interruptions once they have been created because it would likely result in confusion for the students. For example, if a mentor sends an announcement interruption to a group of five students, and three out of the five immediately interact with the announcement interruption, but the remaining two choose to ignore it until the bounded deferral timer expires, then deleting the sent announcement interruption would confuse the three students who already read it. They would read the announcement and internalize the information, and the two students who did not read the announcement interruption yet would have no knowledge of it as it would be deleted before they are able to acknowledge it. As such, editing and deleting interruptions can easily result in confusion for the students.

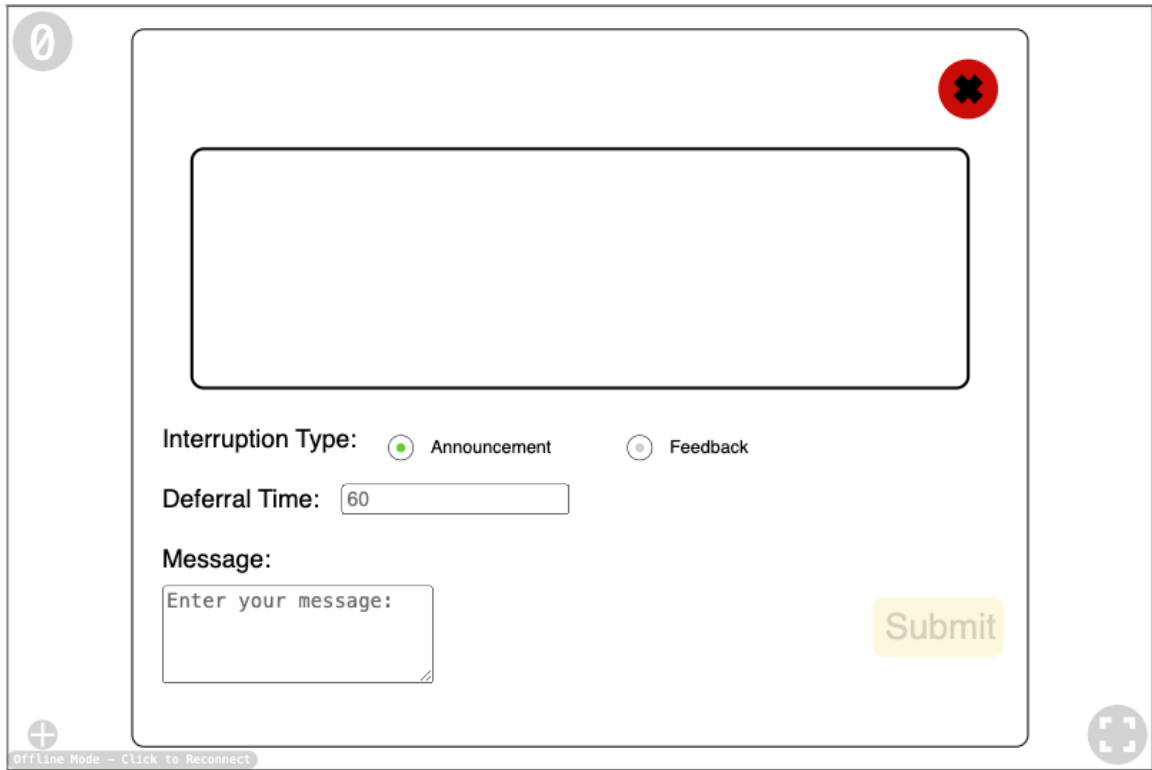


Figure 5.2: A screenshot of the Teacher Screen in STaBL.rocks after clicking the Teacher button on the homepage.

5.2 Student Screen

A user can navigate to the Student Screen by entering a unique user id and then clicking on the “Student” button on the homepage. Once inside the Student Screen, if there are no notifications then there will be a happy face on the screen, as seen in Figure 5.6. It is important that the user id entered is unique, because notifications are filtered and tracked at the user id level, and any non-unique user ids will result in erroneous behaviour in the program. The IMS prototype integration into the Teacher Dashboard and MultiShapeCreator will not require the student to enter a unique id, because the student must login to access MultiShapeCreator, and their unique

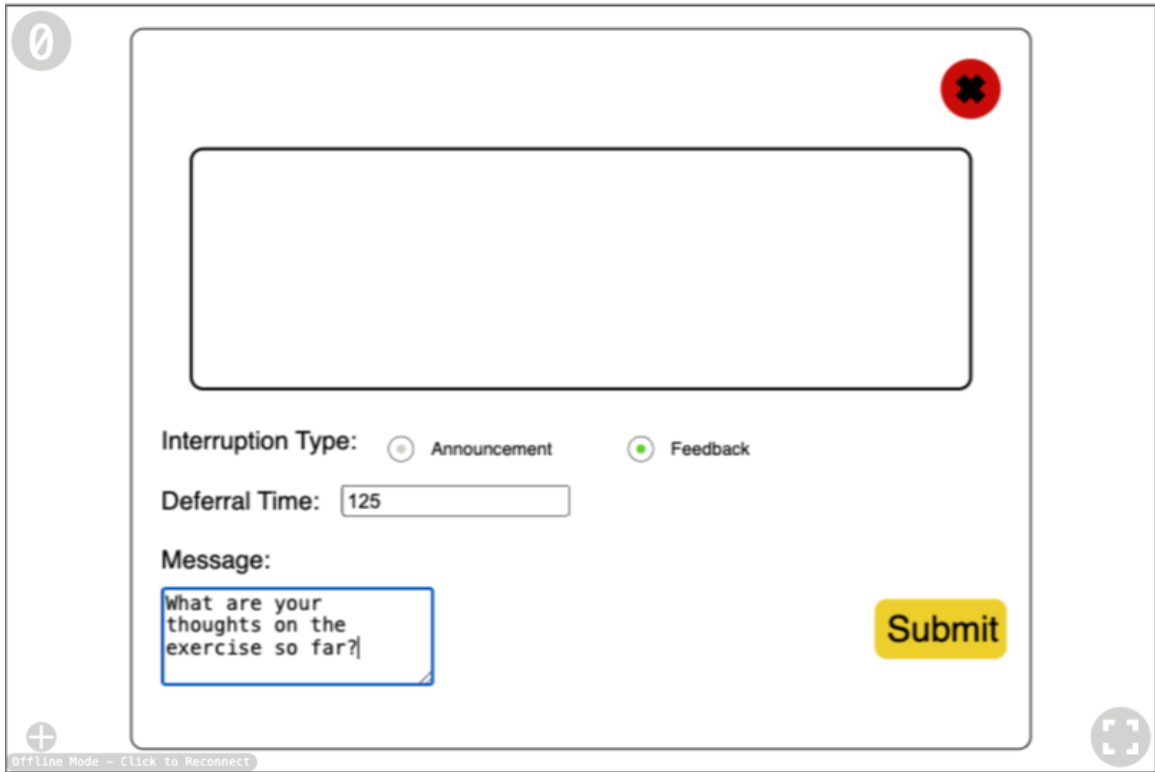


Figure 5.3: A screenshot of the Teacher Screen in STaBL.rocks after clicking the Teacher button on the homepage and entering the inputs to create an interruption with the interruption type Feedback.

MultiShapeCreator login id will be used to track interruptions.

Once an interruption has been created in the Teacher Screen, there will be a notification symbol that appears on the happy face on the screen. The notification symbol is an indicator that there is one or more notifications that require the user's attention; a screenshot of the happy face with a notification symbol is in Figure 5.7. The happy face will slowly change to a neutral face and eventually a sad face if the notification hasn't been attended to, as the bounded deferral time for the notification is approached. An example of the happy, neutral and sad faces are in Figure 5.9. The decision to personify the software tool in the form of a happy, neutral, and

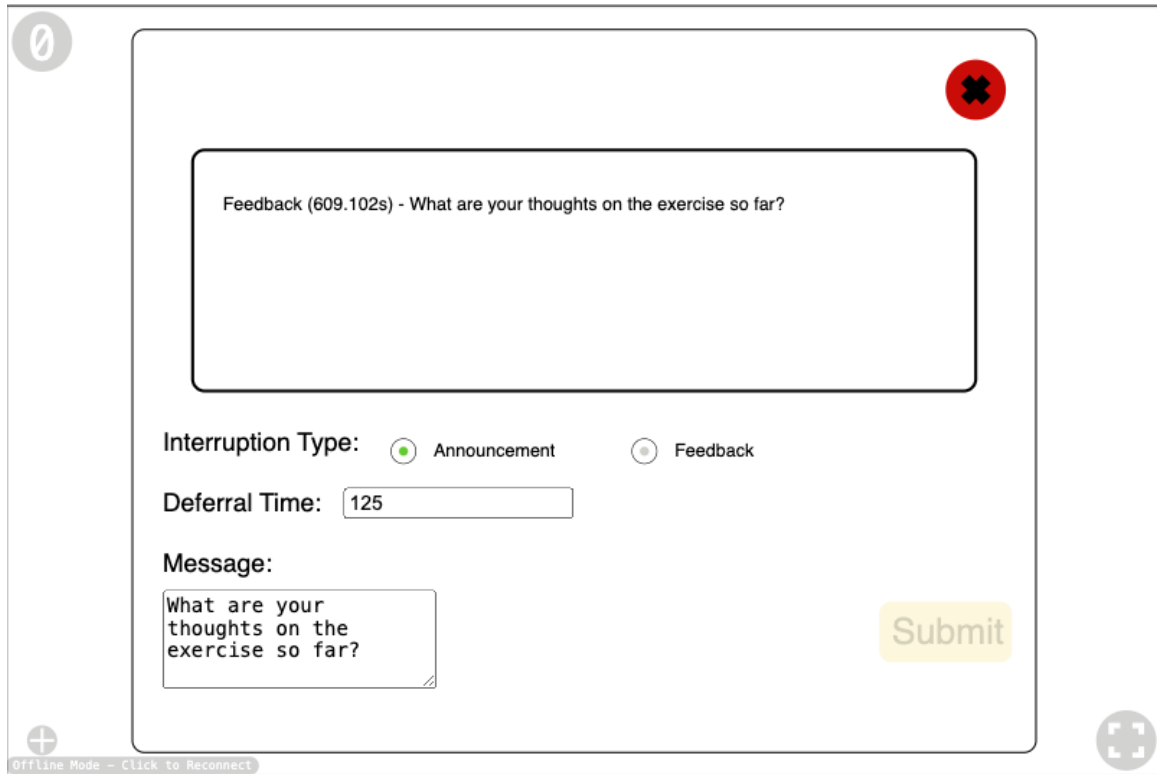


Figure 5.4: A screenshot of the Teacher Screen in STaBL.rocks after clicking the Teacher button on the homepage, and creating an interruption.

sad face was inspired by existing literature, discussed in the literature review, which identifies that there is a higher student engagement when software tools are personified [26]. A screenshot of the three face expressions without the notification symbol is in Figure 5.5.

Once the notification screen has been opened, the student will either “Acknowledge” an Announcement (Figure 5.8) or enter their comments for a Feedback interruption type. Moreover, a student will interact with a notification in one of three ways: (a) selecting a feature, (b) intentional notification selection, and (c) bounded deferral expiry. As such, a user can be interrupted in the aforementioned three ways

as well. Further, each of these methods is inspired by the user interruption resolutions outlined in McFarlane’s Taxonomy. Selecting a feature, which has been designated as an *opportune* feature by a developer, is a mediated interruption since the developer is the proxy applying their own expertise to each feature. Selecting the happy face intentionally to trigger the notification screen is a negotiated interruption because the user has the opportunity to delay the interruption until a further time if they choose to do so. Finally, once the bounded deferral timer expires then the notification screen will be triggered, which is a scheduled interruption.

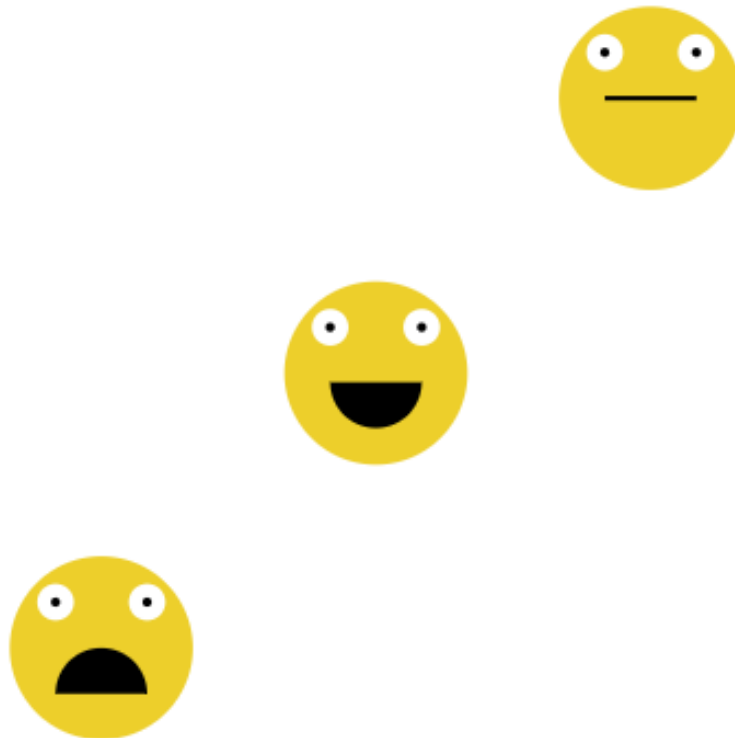


Figure 5.5: The personification of a software tool, where the state of sadness (bottom right), happiness (middle), and neutrality (top right) is demonstrated through the use of smiley faces.

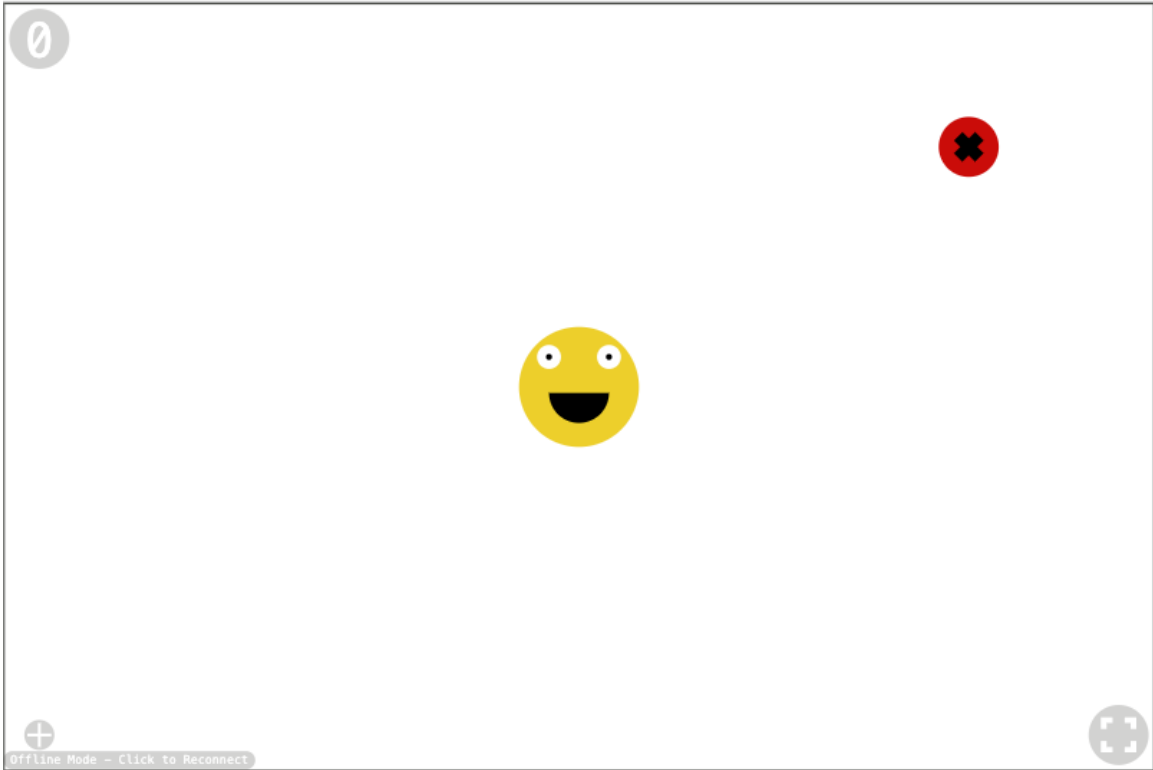


Figure 5.6: A screenshot of the Student Screen in STaBL.rocks after clicking the Student button on the homepage.

5.3 Interruption Classifications

McFarlane’s Taxonomy [33] was applied to the prototype to help evaluate and validate the impact of the interruptions on the students that will be receiving the notifications. McFarlane’s Taxonomy was not directly used to inform the design of the Interruption Management System (IMS) prototype, but it was useful in the evaluation and classification of the prototype. It offered a systematic framework to determine the source, nature, and effects of the interruption from the bounded deferral timer, which played a valuable role in validating the prototype’s behaviour and its intended interruption effects.

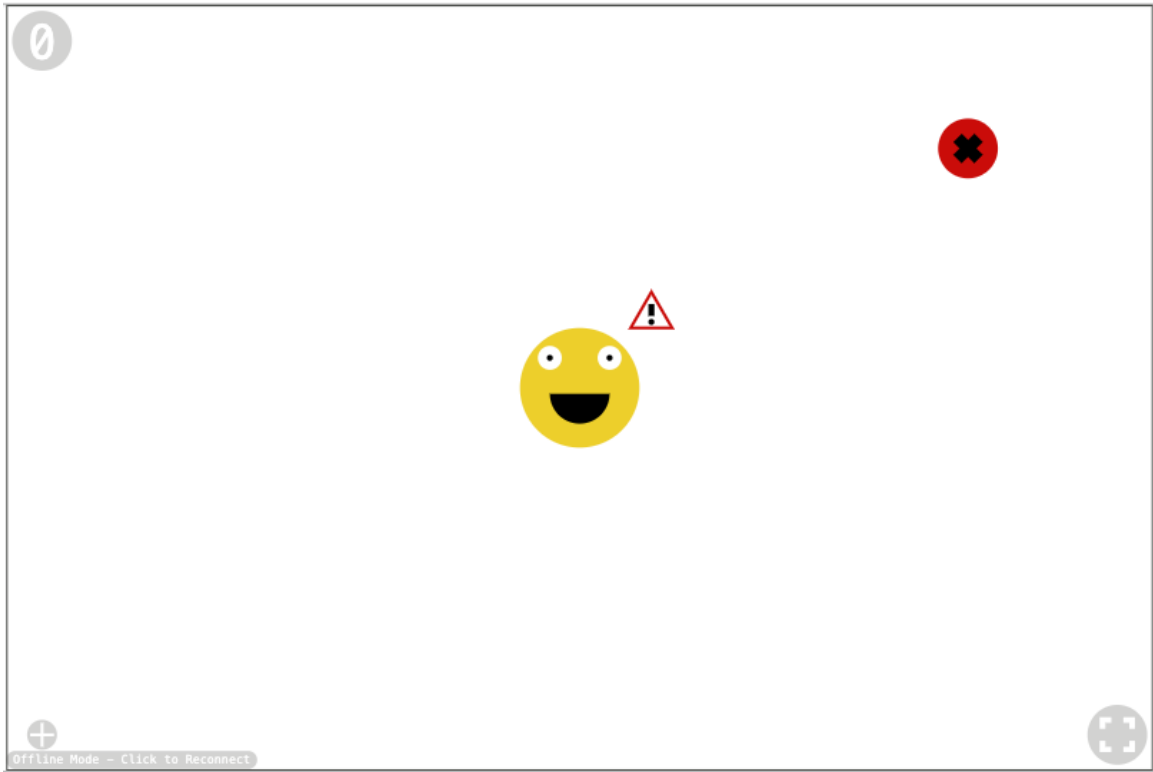


Figure 5.7: A screenshot of the happy face with a notification symbol on it, indicating there are notifications that the user needs to attend to.

A detailed review of McFarlane’s Taxonomy is available in Chapter 2.

- **Source of Interruption:** The source of the interruption will be a notification from STaBL.rocks while students are coding on the integrated development environment.
- **Characteristics of the User Being Interrupted:** Typically, the users that will be interrupted will be students that are using the MultiShapeCreator because STaBL Foundation typically engages with K-12 students.
- **Coordination Method:** If the interruption type is *Announcement* then the



Figure 5.8: A screenshot of the Notification Screen when it is presented to the user. In this case, the Notification Screen contains an Announcement.

user has to click an *Acknowledge* button, however if the interruption type is *Feedback*, then the user will have to enter their comments and click a *Submit* button.

- **Interruption Meaning:** The purpose of the interruption is set by the teacher (or other admin user), however it can be broadly defined in two categories: (a) Announcement which is read-only, and (b) Feedback which allows the receiver to submit comments.
- **Expression Method:** The interruptions will be presented to the user inside the STaBL.rocks interface, inside of a Notification Screen outlined in Figure 5.8

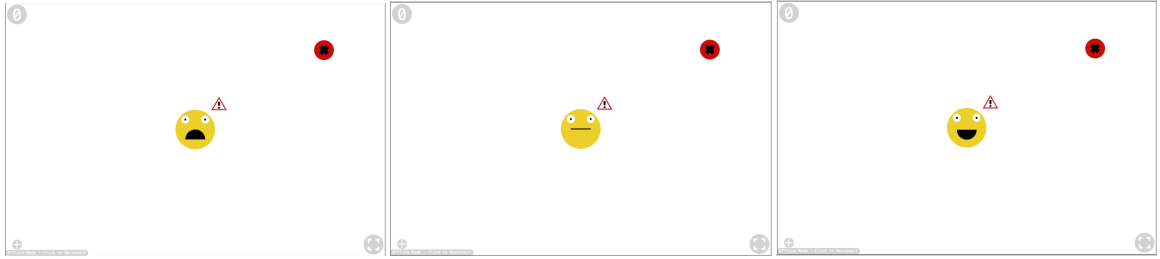


Figure 5.9: Three individual screenshots of the bounded deferral timer expiring. The face is happy when there is a lot of time left on the timer (right), the face becomes neutral when there is less time left (middle), and the face becomes sad when there is close to no time left (sad). The condition for determining how much time on the bounded deferral timer corresponds to each face is set by the developer.

- **Channel of Conveyance:** The channel of conveyance will be the STaBL.rocks platform, and the Notification Screen.
- **Human Activity Changed by Interruptions:** Depending on the notification that’s delivered, the user will have to select the *Acknowledge* button or enter their comments in the Notification Screen at the very least.
- **Interruption Effect:** The prototype was built to minimize interruption to the user by only interrupting the user during opportune moments. It does this by interrupting the student when the bounded deferral timer has expired, or when the student interacts with an opportune feature, which is defined by hierarchical task analysis (discussed in Chapter 2).

5.4 Requirements Delivery

This section focuses on how the prototype meets the requirements standards that were defined in Chapter 3. Table 5.1 identifies how each requirement is fulfilled by

specific components within the prototype or the prototype as a whole. Ultimately, all of the initial requirements have been met in the implementation of the prototype.

Table 5.1: Captures the initial delivery requirements for the prototype and determines whether they have been met by the alternative solution.

Requirements		
Requirement	Met or Unmet	Comments
REQID 1: Outline plan to integrate into MultiShapeCreator & Teacher's Dashboard	Met	A plan to integrate the prototype into the MultiShapeCreator and Teacher's Dashboard, is discussed in the section <i>Seven Stages of Action in the Teacher Dashboard and MultiShapeCreator</i> . A developer will have to effectively use the same paradigm as the prototype and apply it to the components in the MultiShapeCreator and the Teacher's Dashboard. Specific components are outlined and highlighted in Figures to capture <i>opportune</i> features.

REQID 2: Reusable by other applications in STaBL.rocks	Met	The prototype was built with reusability and portability in mind so that it will be reusable by other applications in STaBL.rocks. The Interruption Management System (IMS) prototype will be available as a library that can be imported into other applications in STaBL.rocks.
REQID 3: Support custom functions to trigger notifications	Met	The Notification Screen is displayed on the premise of a state transition, captured by the Model View Update logic in the application. Developers can write custom functions to trigger this respective state change which will trigger the Notification Screen to pop up.

REQID 4: Component library to build interruption management systems	Met	There are various UI components that have been built as a part of the prototype. These UI components can be lifted, shifted and overlaid on top of existing applications in STaBL.rocks. These UI components will further help developers build interruption management systems in existing applications.
REQID 5: McFarlane’s Taxonomy Documentation	Met	Please refer to the subsection <i>Interruption Classifications</i> .
REQID 6: Tag components with Seven Stages of Action	Met	Please refer to the subsection <i>Seven Stages of Action in the Teacher’s Dashboard and Multi-ShapeCreator</i> .

REQID 7: All interruptions must abide to bounded deferral	Met	The <i>Interruption</i> Type contains a field <i>boundedDeferralTime</i> which is a <i>Float</i> type that captures the bounded deferral timer expiry of an interruption. As such, every single interruption in this interruption management system prototype must have a bounded deferral time.
---	-----	--

5.5 STaBL.rocks Proposed Usage Example

A concrete usage example of the prototype is outlined in this section. Its purpose is to illustrate how the prototype would be used by mentors, teachers, and students in a virtual or in-person classroom setting. There are two types of users that are used in this usage example, the first is the mentor. Mentors include teachers and STaBL Foundation mentors, as they effectively participate in the same role in Elm workshops. The other type of user is a student. Students include all of the students who are using STaBL.rocks in an Elm workshop.

A prerequisite to all of the functions below will be to log into the Teacher Dashboard using the appropriate login credentials. Once authorized to use the Teacher Dashboard, the next step will be for each user to navigate to the appropriate tool. For the mentor, the next step will be to navigate to and open the Teacher Screen, whereas for the Student, it will be to navigate to one of the tools they will be using

in the Elm workshop. As such, the mentor will navigate to only the Teacher Screen whereas a student could navigate to many tools. It is assumed that the Interruption Management System (IMS) prototype is integrated into all of those tools, so that the student is reachable via the notification system.

5.5.1 Creating a Notification

The mentor can create two types of notifications in the prototype. It is more straightforward to create an Announcement notification than it is a Feedback notification. An Announcement notification is unidirectional and a mentor simply has to input their Announcement into the Message box and set a Deferral Time. If a mentor wants to collect Feedback, ideally mentors and teachers have a question bank which they can source questions from, to ask the students about their experiences during the Elm workshops. A mentor can enter the question into the Message box and set a Deferral Time. In either case, once the mentor has filled out the form to create an interruption, they must press Submit which will add the Announcement or the Feedback to the queue. In Figure 5.4 the mentor has submitted a Feedback question asking, “What are your thoughts on the exercise so far?” and has set a Deferral Time of 125 seconds.

5.5.2 Responding to a Notification

Once a notification has been received, the student will see the bounded deferral emoji on the bottom-right corner of their screen. It is assumed that the bounded deferral emoji is integrated into all of the tools on STaBL.rocks. The emoji will have a notification on it, as seen in Figure 5.7 but will slowly begin to change its expression,

the longer the notification is not answered. Once the bounded deferral timer that has been set by the mentor expires, then the student will be prompted to answer the question or acknowledge the announcement. Figure 5.8 is a screenshot of what the student would see when they must acknowledge an announcement. The student will not be able to resume their work until the notification has been addressed. Figure 5.9 represents how the bounded deferral emoji will change its expression as the timer begins to expire. After the student has answered a question, the response should be stored in a permanent data store for further review.

5.6 Types

In this subsection, the types are listed such that it outlines how the program functions and the various states that the program can be in, at any given time.

Local Message

The Local Message is responsible for tracking all of the local events that are happening on a user's screen, such as tracking when a user has opened the screen and has entered a response. Since there are two different types of primary users for this program, the teacher/mentor and the student, the local messages can be categorized as such as well.

The local messages `OpenTeacherScreen`, `CloseTeacherScreen`, `EnterITChange`, `EnterMsgChange`, `EnterBDChange`, and `ResetTeacherScreen` are all tracked for the teacher, whereas `EnterStudentId`, `OpenStudentScreen`, `CloseStudentScreen`, `OpenStudentNotification`, `EnterStudentFeedback`, and `CloseStudentNotification` are all tracked for the student.

The `OpenTeacherScreen` message is responsible for changing the state to open

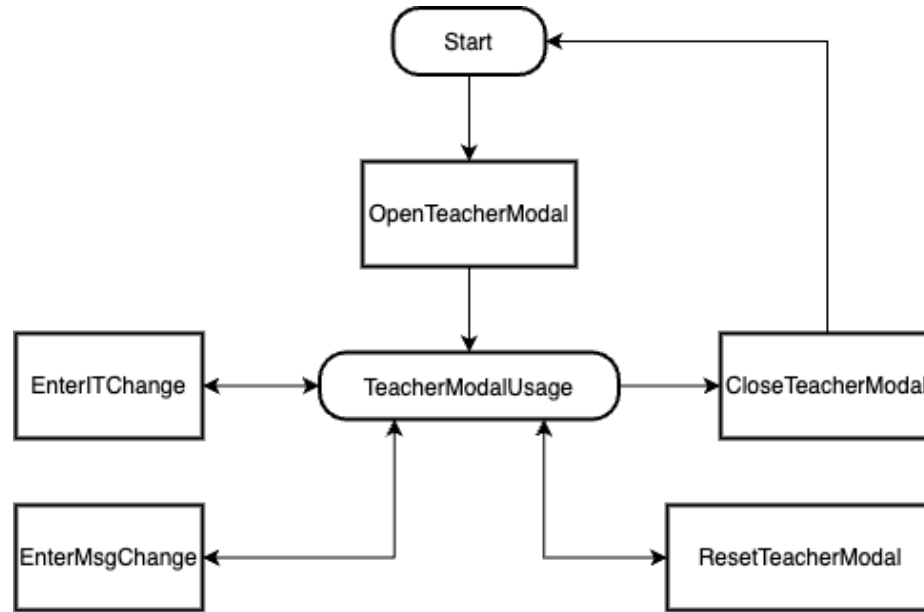


Figure 5.10: Teacher UI state diagram outlining the various states that a user that has the teacher role can be in.

the teacher screen, and the `CloseTeacherScreen` message is responsible for changing the state to the start screen from the teacher screen. The other four messages, `EnterITChange`, `EnterMsgChange`, `EnterBDChange`, and `ResetTeacherScreen`, are responsible for entering data about the notification that the teacher or mentor wants to create. `EnterITChange` is responsible for tracking the local state between two radio buttons determining whether the notification delivered will be an announcement or will require feedback; a user can select either one, and when they click on a radio button, then the `EnterITChange` message will be sent to trigger the change in local state. `EnterMsgChange` is responsible for tracking the local state of the text that is to be delivered with the notification, each keystroke is captured in the local state. `EnterBDChange` captures the amount of time the notification should adhere to a maximum bounded deferral. The `ResetTeacherScreen` will reset the values of the

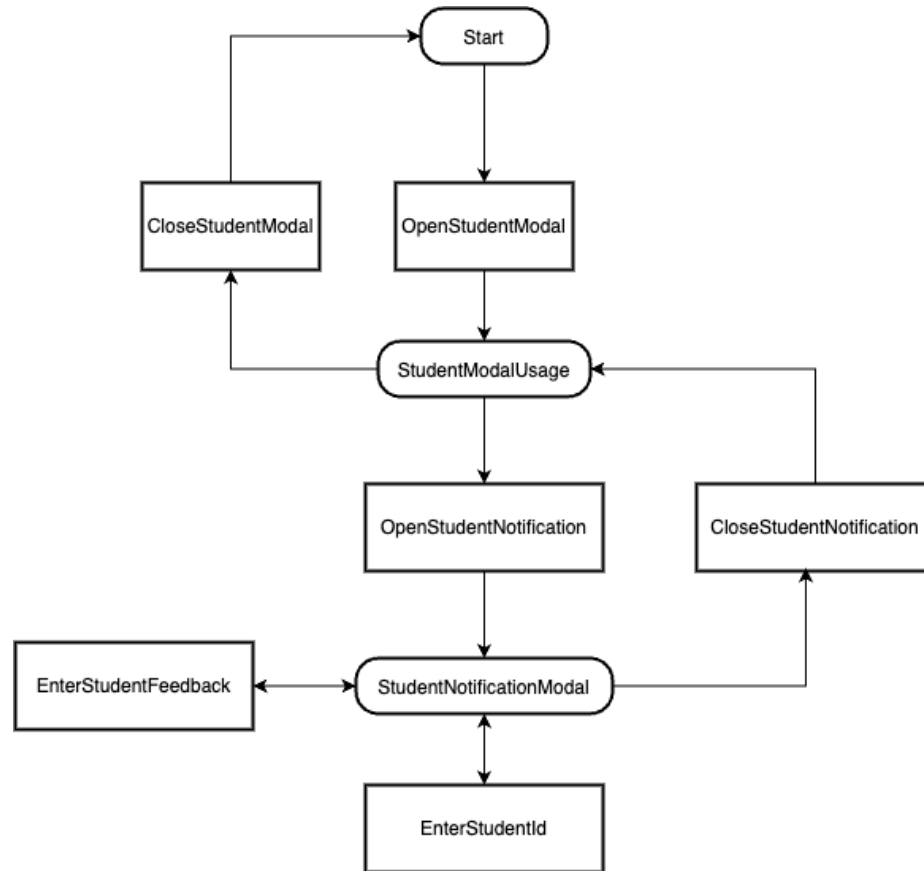


Figure 5.11: Student UI state diagram outlining the various states that a user that has a student role can be in.

local state back to their default values, and is typically used once a notification has been submitted to the global model. Figure 5.10 illustrates the states and transitions that a mentor can be in while using the IMS prototype. Similarly, Figure 5.11 illustrates the states and transitions that a student can be in while using the IMS prototype. The Elm code that was written to define the messages (transitions) and states is listed in Appendix B.

Chapter 6

Results of Research Cycle #1

6.1 Feedback Collection Setup

The first iteration of the prototype was developed using an initial set of requirements that originated from an ideation session with the Managing Director (MD) of STaBL Foundation. To determine whether the developed prototype was useful, four semi-structured interviews were organized with STaBL Foundation MD and mentors. The semi-structured interviews were conducted using Microsoft Teams¹, which is a video conferencing tool.

A demo of the prototype was prepared for each of the virtual calls to demonstrate its capabilities. After establishing rapport, as discussed in Chapter 3, each participant was shown the teacher and student screens to demonstrate the functionality of the prototype. Specifically, an “Announcement” interruption type was created on the teacher screen and queued up for 60s before navigating to the student screen and interacting with the personified bounded deferral timer emoji. The same process was

¹<https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>

repeated for the “Feedback” interruption type. The concept of the bounded deferral timer was also explained to each of the participants. During the demonstration, the participants were encouraged to ask questions about the functionality. Once the demonstration was completed, the participant was asked the guiding semi-structured interview questions, which are listed and discussed in Chapter 3.

6.2 Feedback Components

The feedback collected from each of the virtual calls has been organized into a general feedback section and respective user interface components that were suggested by the participants.

6.2.1 General Feedback

The general feedback that was received from each of the mentors and the MD was positive. Ultimately, all four participants thought that the tool would be useful to communicate with students in a virtual or in-person classroom. Participants mentioned that feedback was rarely collected from students and it was difficult to gauge the feelings of a student during or after the instructional sessions. All of the participants indicated that the bounded deferral timer personification was a good method to politely grasp the student’s attention without being disruptive. Further, all participants indicated that the personified emoji should be transparent and tucked at the bottom of the page, where it is unobstructive for ongoing tasks. Placing the personified emoji in the center of the page, or somewhere else which is obtrusive, will immediately capture the student’s attention and will result in higher disruption.

Two mentors suggested that the “Announcement” functionality is not useful in an in-person classroom because mentors can yell announcements in the classroom, however randomly timed announcements may result in inopportune interruptions that disrupt the student from their primary task. The usefulness of the Feedback Collection functionality was further solidified through positive affirmations from each participant. Each participant indicated that a systematic database of questions can be used to capture useful feedback about a student’s feelings in a non-disruptive manner using the prototype.

6.2.2 Capture Attention Differently

Multiple participants suggested that the personified emoji changing emotions was not disruptive enough to capture the attention of students. Two features were proposed to enhance the personified emoji: (a) a message bubble on the emoji, and (b) the emoji raising hands and having “jitters”. The feedback about the existing emoji was that it was not clear why the emoji was becoming more upset over time. Unawareness of the cause of the emoji’s sadness could result in feelings of frustration in students because they do not understand what they are doing to elicit such a reaction.

The message bubble enhancement was proposed because it indicates that the emoji has something that it wants to say to the student. A small message bubble would overlay the emoji so that it would look like the emoji has something on its mind. This subtle change would communicate the intent of the emoji and naturally imply why it is becoming more “upset” over time. Further, if there are multiple messages sent by mentors or teachers, then they can be displayed through multiple message bubbles on the emoji up to a maximum number.

Another proposed enhancement is to instill the qualities of a young student in the emoji. With this proposed enhancement, the emoji would have an animated hand beside it which would slowly rise as the bounded deferral time expired, and instead of becoming sad it would continue to stay happy but would start to vibrate. The vibration and raising of the hand is meant to copy the characteristics of a young student who is very excited to answer a question or share something in the classroom. Younger students may find this enhancement more relatable because it is something that they also experience or have commonly seen other students do.

6.2.3 User Roles

Multiple participants suggested that there should be different tiers of roles in the next iteration of the prototype. They suggested that an administrative role be added to the prototype to capture which students should be prompted for feedback and which should not. Naturally, the majority of users of the student role in the prototype are students who are learning Elm programming, however sometimes mentors are also participating in coding activities. While it may be argued that they should, mentors aren't expected to be answer feedback questions to gauge how they feel while programming on STaBL.rocks. The administrative role would enable the ability to programmatically exclude users from the feedback questions or announcements. Other roles suggested would include enhanced student roles which would enable students with privileges to ask questions or make announcements to a subset of students. The logic behind this idea is that there may be "team-leads" while students are learning Elm programming that would want to gauge their team's feelings. Another use case is when students are engaged in a hackathon-style competition and want to announce

interesting features they've developed, especially if they are working asynchronously. Typically, students in a classroom are able to communicate verbally with each other but the facilitation of communication in an asynchronous setting is possible with the enhanced roles.

Participants also suggested that students, and their role, should be able to be easily identified through the prototype. The prototype should use a custom avatar that the student creates and assign it a signifier if it is an enhanced role, such as a star. Using custom avatars would help teachers, mentors, administrators, and enhanced-privilege students identify students much more easily, rather than the alternative of relying on the username text.

6.3 Prototype Alternatives

The only alternative to collect feedback, which was identified through conversations with the Managing Director of STaBL Foundation, is the use of Google Form and Google Meet.

Google Form is a form creation and hosting tool developed by Google that allows users to create forms and send them to other individuals. The following types of questions can be asked in Google Forms:

- Short Answer
- Paragraph
- Multiple Choice
- Checkboxes

- Dropdown
- Linear Scale
- Multiple Choice Grid
- Checkbox Grid
- Date
- Time

Mentors would create Google Forms that sourced questions from the Question Bank in Appendix A, and ask students to fill out the questions after an in-person or virtual workshop/session. Typically, only the Short Answer and Multiple Choice question types were used in the forms. Google Meet is a video communication service which can be used to host virtual meetings, and it is what the mentors used to conduct any in-person interviews to collect feedback from students.

The alternative of using Google Form and Google Meet, unfortunately, does not meet a lot of the requirements that were derived from the motivating problem. Table 6.1 captures which requirements are (un)met in the alternative solution and naturally outlines a greater need for the prototype for the requirements which are unmet.

Table 6.1: Captures the initial delivery requirements for the prototype and determines whether they have been met by the alternative solution.

Requirements		
Requirement	Met or Unmet	Comments

REQID 1: Outline plan to integrate into Multi-ShapeCreator & Teacher's Dashboard	Unmet	Solution does not natively integrate into STaBL.rocks
REQID 2: Reusable by other applications in STaBL.rocks	Unmet	Solution does not natively integrate into STaBL.rocks
REQID 3: Support custom functions to trigger notifications	Met	Functionality available to trigger custom notifications
REQID 4: Component library to build interruption management systems	Unmet	N/A does not support interruption management systems
REQID 5: McFarlane's Taxonomy Documentation	Unmet	Development of Taxonomy Documentation required.

REQID 6: Tag components with Seven Stages of Action	Unmet	Tagging components in STaBL.rocks will not be able to trigger opening of Google Form
REQID 7: All interruptions must abide to bounded deferral	Unmet	N/A interruption via Google Form is facilitated manually by mentors

Chapter 7

Prototype Iteration #2

The second iteration of the prototype focused on implementing the features that were suggested in the first round of semi-structured interviews. There were several features that were suggested, however the following three were most frequently cited by mentors:

- Capturing attention differently with updates to emojis
- Updating user roles to contain admin roles
- Identifying users through custom avatars

As discussed in Chapter 3, the next step in the Design Thinking process is to incorporate the suggested features into the prototype. The primary feature that was implemented, was to capture attention differently with updates to the bounded deferral timer emoji. The other two features-admin roles and custom user avatars-were not integrated into the IMS prototype standalone application, but are supported by the architectural components in the Teacher Dashboard (see Chapter 4). The standalone

prototype was not integrated with the Teacher Dashboard due to significant developer rework required, but the identification of the feature availability in the Teacher Dashboard is necessary for future studies where the IMS prototype will be integrated into the Teacher Dashboard.

The focus of this design iteration was to implement *jittering* hands and the thought bubble in addition to the existing sad, serious, and happy faces in the bounded deferral time emoji. Mentors were concerned that the emoji in its current state would not be able to capture the users' attention and any subtle attempts to capture the user's attention would be unnoticed or result in frustration. The *jittering* hands and the thought bubble added more contextual clues to the emoji to highlight why the emoji is interrupting the user.

7.1 Implementation

The implementation to update the emojis was completed using the STaBL.rocks IDE. The implementation for both features was straightforward, as all of the development was contained within the IDE.

7.1.1 Thought Bubbles

Figure 7.1 is a screenshot of the implementation of the thought bubbles added to the emoji in STaBL.rocks. The implementation required the creation of several UI components to add the thought bubbles to the emoji. The thought bubbles were not modeled after a specific animated picture, but were drawn from memory from commonly seen thought bubbles in many animated movies.

7.1.2 Jittering Hands

Figure 7.2 is a screenshot of the implementation of the *jittering* hand added to the emoji in STaBL.rocks. Alongside the implementation of several new UI components, the *jittering* aspect of the feature required animation to be added. Naturally, motion cannot be captured in a screenshot so it is not available to be seen in Figure 7.2, however its implementation is further defined in this subsection.

This code block outlines the implementation of the emoji in the prototype. `baseFace` is a variable which is the base of the emoji's face. Other features are added to the `baseFace` such as the expression and paw. Naturally `smile`, `frown`, and `netural` capture the expressions in each emoji. `paw` is a variable for the UI elements which comprise the emoji's paw. The animation is determined in the `move(15, 10*sin(model.time))` function which instructs the paw to move up and down, since `sin` is a cyclical function. The input of `model.time` enables the cyclical movement as it is updated with each `Tick` in the application, corresponding to UNIX time. The extent of the movement of the paw can be controlled by the amplitude of the sine function which is currently set to 10.

```
face emotion model =  
  if emotion == Happy then  
    group [  
      baseFace,  
      smile |> move (0, -1),  
      paw |> move(15, 10*sin(model.time))  
    ]  
  else if emotion == Sad then
```

```
group [  
    baseFace,  
    frown,  
    paw |> move(15, 10*sin(model.time))  
]  
else  
group [  
    baseFace,  
    neutral,  
    paw |> move(15, 10*sin(model.time))  
]
```

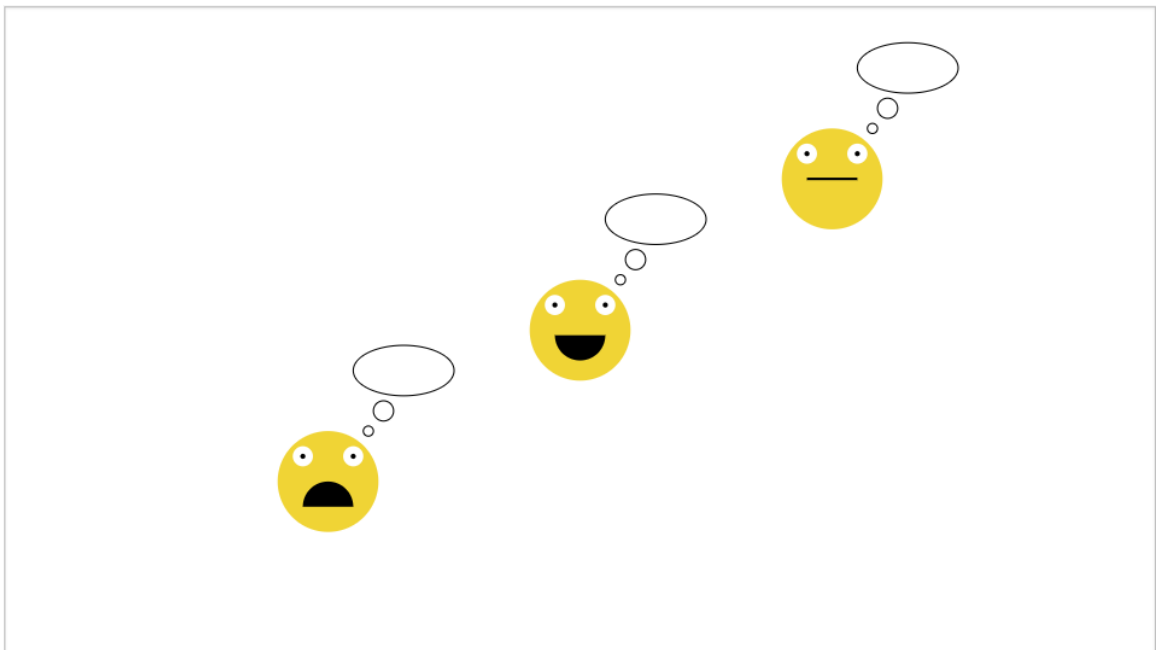


Figure 7.1: A screenshot of a potential enhancement to the emojis capturing *thought bubbles* being added to the emoji which would help capture the user's attention better.

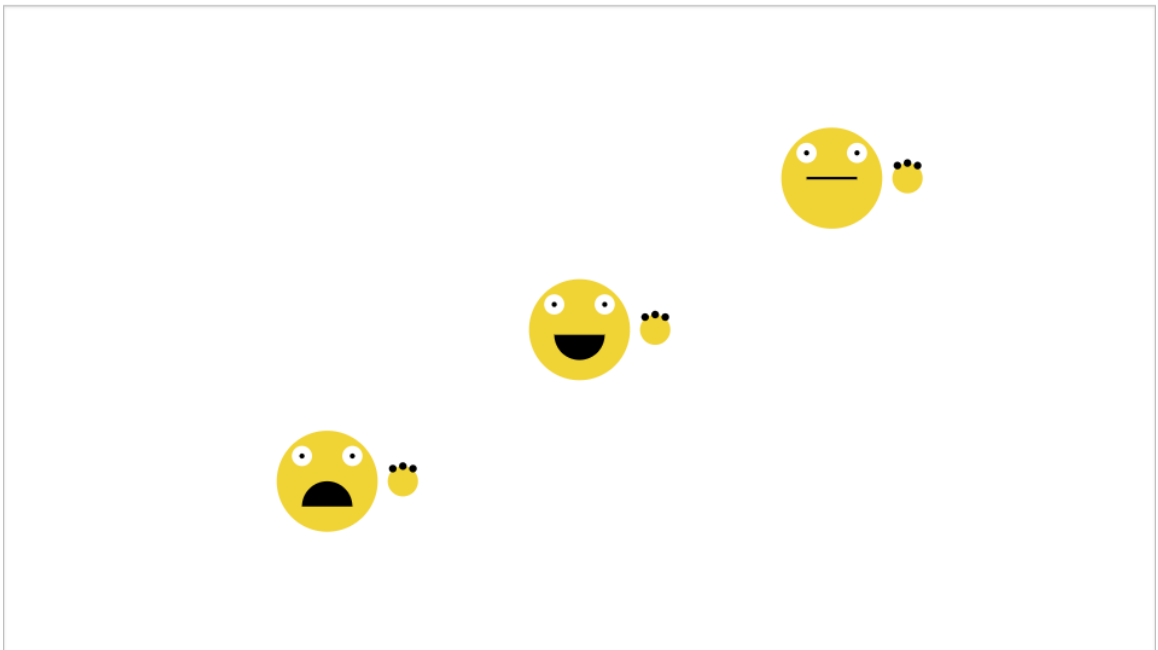


Figure 7.2: A screenshot of a potential enhancement to the emojis capturing *paws* being added to the emoji which would raise up and down to capture the user's attention better.

Chapter 8

Results of Research Cycle #2

This chapter presents the findings from the second research cycle for the Interruption Management System (IMS) prototype. Two mentors in the STaBL Foundation participated in individual semi-structured interviews to provide feedback about the second iteration of the IMS prototype, which was developed after Research Cycle #1 (detailed in Chapter 7). The second iteration of the prototype integrated the “jittering hands” functionality to better capture the students’ attention, in addition to the smile turning to a frown when the bounded deferral timer is expiring. The same semi-structured interview setup from Chapter 6 was used, except the second iteration of the prototype was demonstrated to the mentors.

The two interviewees are referred to as Mentor A and Mentor B in this chapter so that quotes can be logically grouped under their respective source, and to protect the identity of the interviewees. Each semi-structured interview lasted approximately 15 minutes, and concluded once both mentors had confirmed that they had no more suggestions.

To determine a new set of features to implement in the IMS prototype, thematic

analysis was applied to the transcript from each semi-structured interview. Both transcripts were reviewed, and interviewee statements were manually coded. Short phrases that communicated feedback for the same user interface (UI) component were coded together. Excitement, future usage, and emotional responses for the IMS prototype were coded separately, and grouped together as “General Feedback” in Section 8.1, as they do not directly inform the development of a new feature. Otherwise, codes were grouped into categories to create overarching themes that informed new features to implement in the IMS prototype, and is further discussed in Section 8.2. The implementation details of the new UI features, resulting from the themes generated by thematic analysis, for the IMS prototype are discussed in Chapter 9.

8.1 General Feedback

Both mentors communicated sentences and phrases that didn’t map directly to new and specific user interface (UI) features but provided affirming contextual information about the Interruption Management System (IMS) prototype. Both mentors expressed interest in using the prototype when visiting classrooms and to specifically use the prototype to collect feedback from students. Mentor A said, “100% I will use this tool. I think every mentor should use it.” Mentor B said, “Everything looks good, I’m excited to use it in a classroom the next time I visit.” Statements from both mentors indicate that they find the IMS prototype useful and will use it to collect feedback from students. Positive feedback indicates that the second iteration of the prototype is meeting the needs of mentors.

8.2 Themes

This section discusses the themes that were determined from the semi-structured interview transcripts. Each theme is presented with corresponding codes and supporting quotes from the interviewees. The first theme is “Announcements Visibility” and refers to the announcement interruption not being visible after the student has acknowledged it. The second and last theme is “Personified Emoji Appeal” and refers to the design and realism of the personified bounded deferral timer emoji.

While only two themes were identified, it is an appropriate number for only two semi-structured interviews that lasted approximately 15 minutes each. Each interview naturally came to a close because the participants had no more suggestions. The presence of only two themes may also indicate that the existing feature set is beginning to “saturate” because there is not as much feedback being provided. Saturation of feedback implies that there is a diminishing amount of value in each design iteration since there is not as much new information being learned about the existing feature set of the IMS prototype. Further, as the sole coder of the transcripts, there is a limitation of subjective interpretation however efforts were taken to minimize any subjective biases while coding by reviewing the transcripts multiple times.

8.2.1 Announcements Visibility

Both mentors raised concerns about the visibility of announcements once a student had acknowledged them. In the current design of the prototype, students cannot revisit announcements once they acknowledge them, which led to concerns about students accidentally acknowledging announcements without understanding its content. Supporting quotes include:

- “Can students see announcements if they **accidentally click acknowledge?**”
- Mentor A
- “...students will **click acknowledge if they are in a rush..**” - Mentor B

Supporting codes include:

- Student acknowledgment behavior
- Announcement visibility

The statements by both mentors indicate that IMS prototype is unable to support students who are interrupted at an inopportune moment by an announcement, or who accidentally click on the acknowledge button. Mentor B specifies that students may be in a “rush” indicating that it is an inopportune moment to interrupt the student.

To further support students who have been interrupted at an inopportune moment, or have accidentally acknowledged an announcement we will add a notification screen that students can access by clicking on the bounded deferral timer emoji, which contains a historical log of announcements. Any student that hurriedly or accidentally clicks the acknowledge button will have the opportunity to revisit missed announcements. As a result, the historical log of announcements was added to the next iteration of the prototype.

8.2.2 Personified Emoji Appeal

Both mentors provided feedback about the UI features of the IMS prototype. Concerns raised included the visual appeal of the IMS prototype to children, and specifically the colour of the IMS prototype and the design of the bounded deferral timer

emoji. A more “child-friendly” design is required to make the tool more appealing for students, who are children, to use.

Supporting quotes include:

- Maybe **add color to the interface** to make it more exciting - Mentor A
- “Especially if it’s kids... we need **something that would catch their attention.**” - Mentor A
- “The **smiley face doesn’t look that smiley** to me ... the **wedge looks weird, a line would look better**, but **I like the paw!**” - Mentor B

Supporting codes include:

- Visual appeal
- Child-friendly design

Statements from both mentors imply that there needs to be significant visual changes to the IMS prototype so that it is appealing to children. Specifically, Mentor A stated that the color of the IMS prototype is not exciting. To make the IMS prototype’s colors more exciting, integrating it into the Teacher Dashboard and MultiShapeCreator, discussed in Chapter 4, will resolve the issue because they both have a consistent styling theme that has been used by students in classrooms. Mentor B communicated that the bounded deferral timer emoji was unappealing to children and needed to be enhanced so that it looked more realistic. Students may not interact with a tool that looks unappealing, so to improve the visuals of the tool, the bounded deferral timer emoji was enhanced. Specifically, an emoji with a realistic smile was implemented into the IMS prototype instead of using the black wedge for its smile.

Chapter 9

Prototype Iteration #3

This chapter discusses the new features that were implemented as a result of the feedback from the second round of semi-structured interviews, discussed in Chapter 8. Specifically, the two features that were identified are to add a historical log of announcements and to improve the design of the personified emoji. The historical announcement log was suggested so that students that accidentally acknowledge an announcement have the opportunity to review it again. The aesthetic changes to the personified emoji were suggested to make the Interruption Management System (IMS) prototype more appealing to K12 students.

9.1 Historical Announcement Log

The “Historical Announcement Log” (HAL) maintains a log of announcements that mentors have sent to students. Students can click on the personified emoji, and click on the “Past Announcements” button to open the HAL. Since Mentors lead different types of instructional sessions with varying instructional time lengths, they are

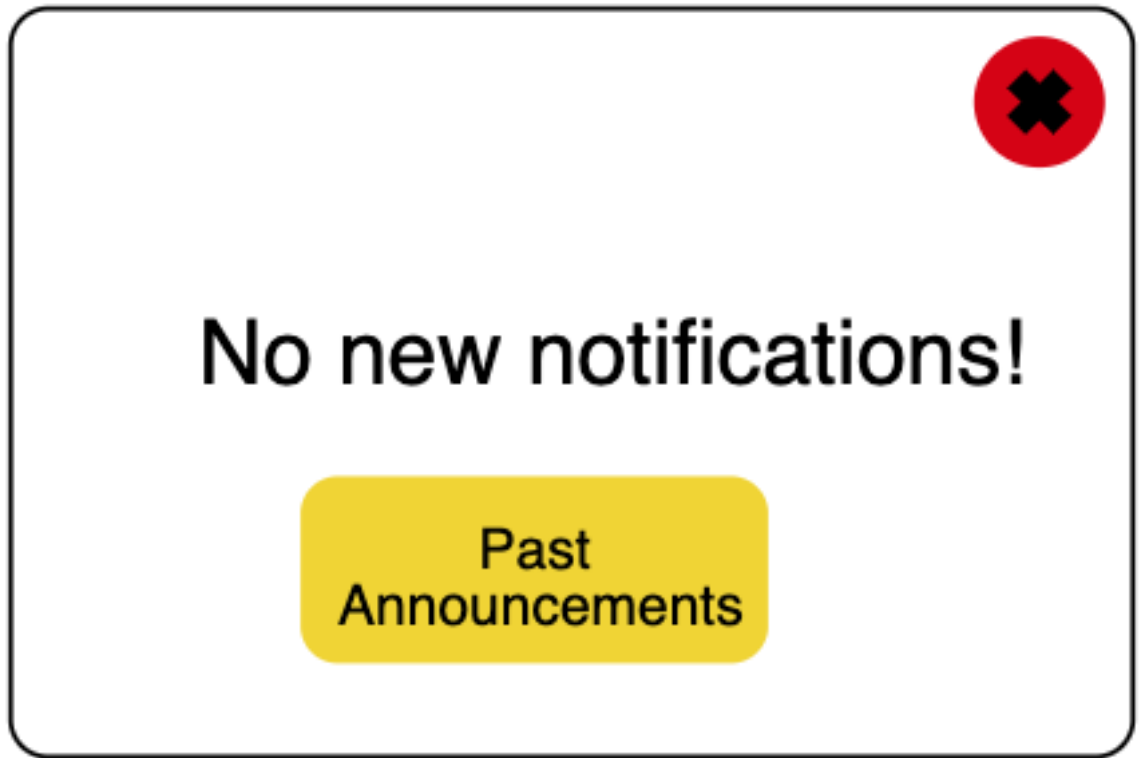


Figure 9.1: A student would click on the personified emoji to open this user interface dialog, and then click on the “Past Announcements” button to open the Historical Announcements Log.

responsible for setting an “expiration time” for announcements in the announcement log. This “expiration time” is different from the bounded deferral timer that Mentors set when creating an announcement. The “expiration time” is used to track how long an announcement should be displayed in the HAL. Once an announcement has reached the predefined “expiration time”, then it will no longer be displayed on the Historical Announcement Log. Mentors enter the expiration time in the Teacher Screen that they use to create interruptions, where the default time for an announcement interruption is 60 minutes. Figure 9.3 is a screenshot of the new Teacher Screen with the “HAL Expiry” input.

The HAL is populated using the existing Global Model InterruptionQueue, which is a queue for interruptions created by Mentors and stored in the Global Model, further discussed in Chapter 3 Section 4.3. To access the HAL, students click on the “bounded deferral timer” emoji and select “Past Announcements”. Once open, students will see historical announcements that they have acknowledged. Figure 9.1 and 9.2 are screenshots of the “Past Announcements” button and the HAL respectively.

9.2 Personified Emoji Aesthetic Enhancements

The personified bounded deferral timer emoji was enhanced by replacing the wedges used for the smile and frown with lines to make the emoji look more realistic. From the semi-structured interviews for the second iteration of the prototype, discussed in Chapter 8, it was suggested by a mentor that this simple change makes the emoji look more realistic and more engaging to students. Figure 9.4 is a screenshot of the new personified emoji with the “line smile” instead of the “wedge smile” that was used in the Interruption Management System Prototype #2.

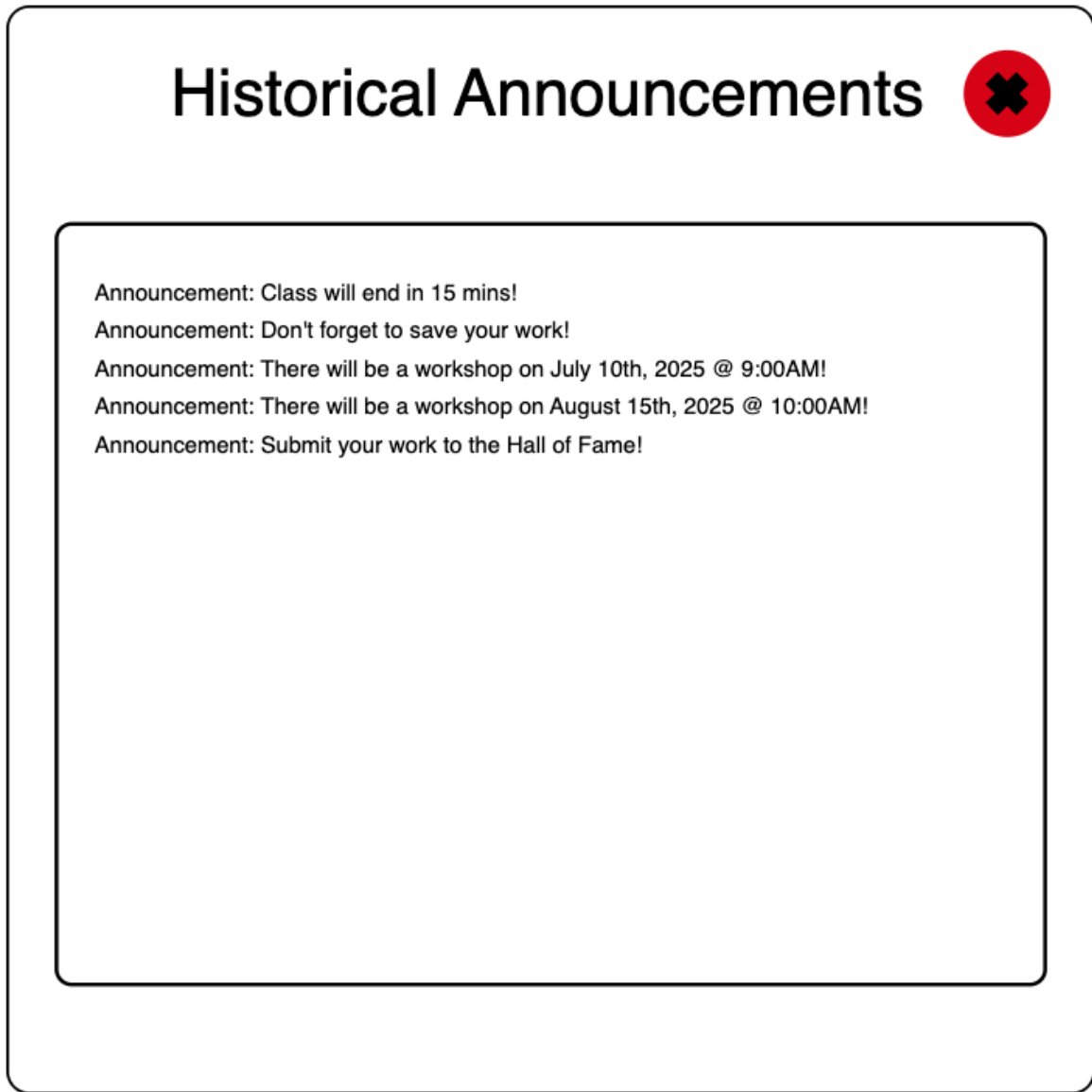
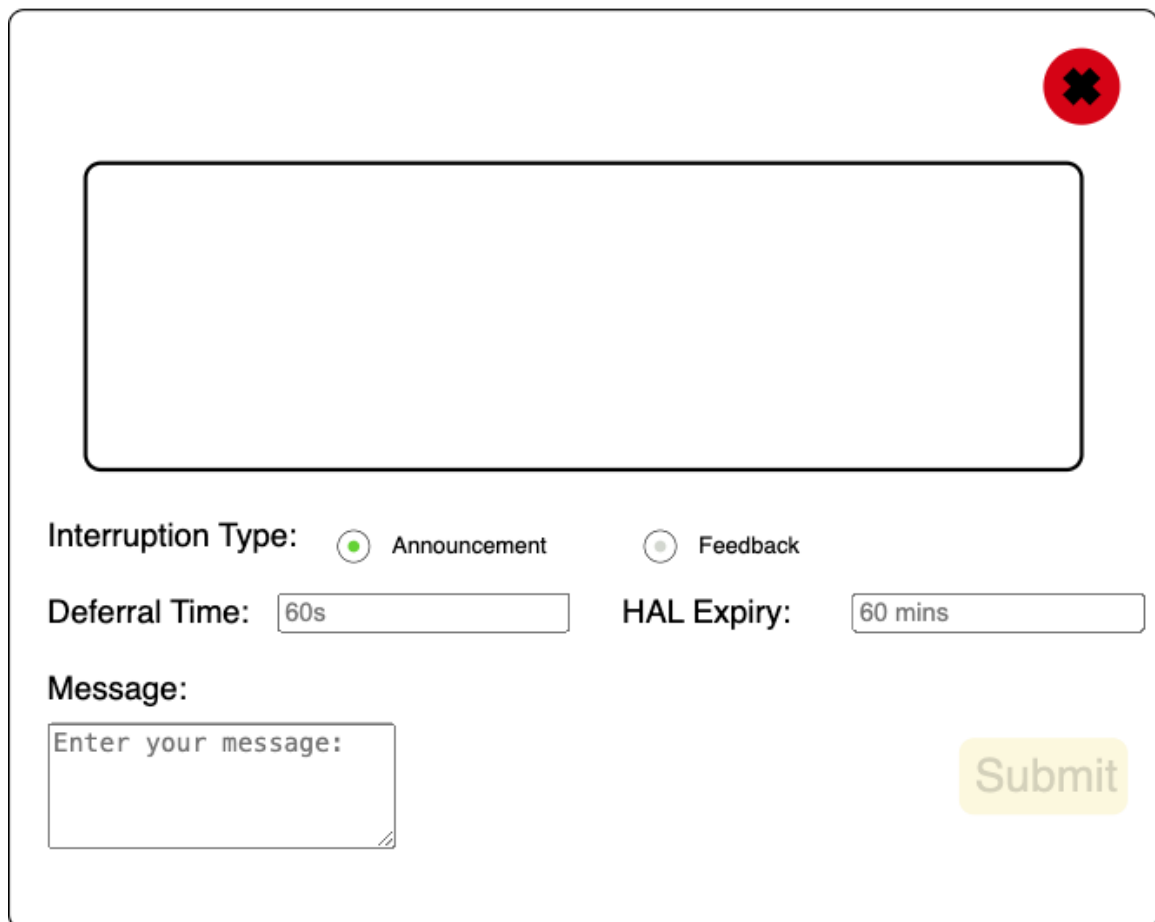


Figure 9.2: Example announcements that a student would see in the Historical Announcements Log.



The image shows a web form for submitting an announcement. At the top right is a red circular icon with a black 'X'. Below it is a large, empty rectangular text area. Underneath the text area are two radio buttons: 'Announcement' (selected) and 'Feedback'. Below these are two input fields: 'Deferral Time' with the value '60s' and 'HAL Expiry' with the value '60 mins'. At the bottom left is a text input field with the placeholder 'Enter your message:'. At the bottom right is a yellow 'Submit' button.

Announcement

Deferral Time: 60s HAL Expiry: 60 mins

Message:

Enter your message:

Submit

Figure 9.3: Mentors can specify the “expiration time” of each announcement they submit.

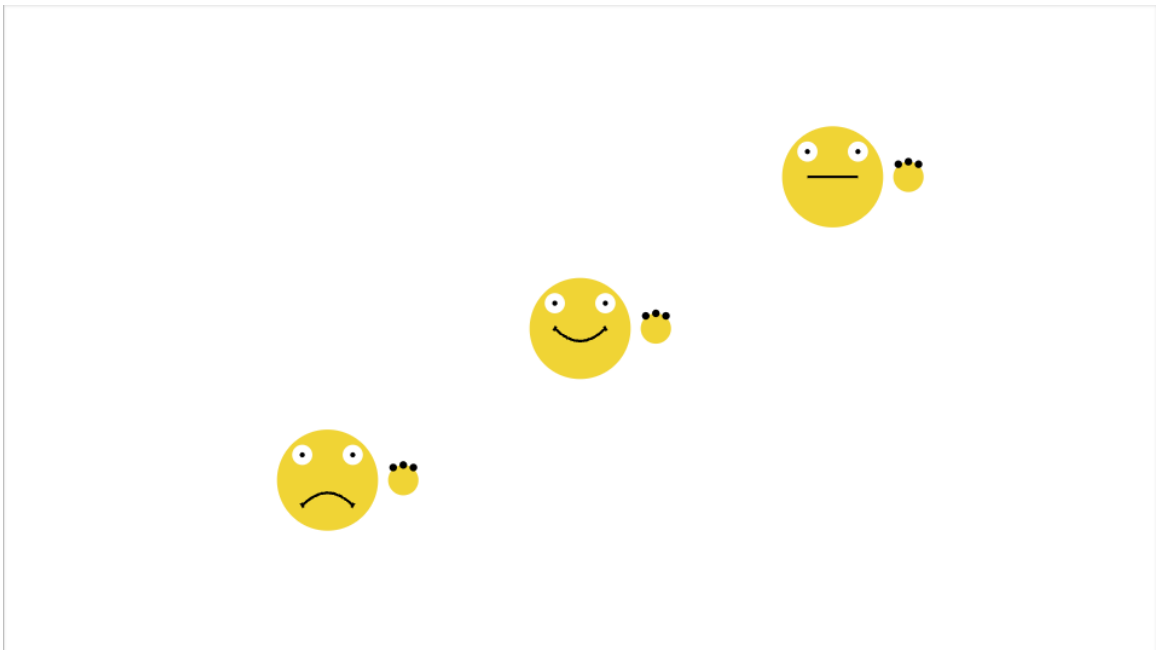


Figure 9.4: The personified emoji for the Interruption Management System (IMS) Prototype #3 with a line for the smile instead of the wedge that was used for the IMS Prototype # 2.

Chapter 10

Future Work

10.1 Gap Analysis

The purpose of the Gap Analysis section is to discuss limitations or new approaches that could be used to conduct novel research for the studies evaluated in the Literature Review Chapter (Chapter 2). The focus of the Gap Analysis will be on the limitations (and opportunities for novel contributions) of Interruption Management Systems (IMS).

10.1.1 Interruption Management Systems

Opportune moments for interrupting users typically occur when users have a low cognitive workload, supported by many empirical studies [9–11]. While the characteristic of an opportune moment is clear, determining when the opportune moment presents itself, is the challenge. For example, Iqbal et al. [22] tracked pupillary movement in an

empirical study to determine when cognitive or mental load was at its lowest, indicating that the user was experiencing a task boundary. Other existing literature aiming to solve this challenge has typically focused on using human speech as an input to determine task boundaries, or directly examining the task structure; the tasks involved in the existing literature typically involve developing a methodology for opportune interruptions from a machine during a human-to-human conversation [42, 41, 3, 37, 47]. Though the existing literature outlines statistical models and methodologies to identify task boundaries using human-speech, the question of whether programming code, specifically Elm code in the context of a web-based integrated development environment (WebIDE), has not been addressed in the existing literature. As such, a novel model to predict task boundaries with programming code as an input can be used to address this need.

10.1.2 Large Scale Studies

Physical equipment requirements, such as eye-tracking sensors or microphones to record human-to-human conversations, have been a barrier to large-scale empirical studies due to budgetary and space constraints [22]. This gap in the existing literature presents an opportunity for the development of a novel methodology for delivering interruptions at opportune moments without the barriers to scaling up. Experiments incorporated into the STaBL.Rocks WebIDE which is used by thousands of learners every year—and growing—could facilitate the administration of a large-scale experiment.

10.1.3 Notification Deferral

The existing literature does not capture a widely accepted or state of the art method to defer notifications to another opportune moment. The existing literature attempts to determine whether the present moment is the most opportune moment to deliver a notification, lacking the knowledge to predict the next best opportune moment to present a notification. Further, machine learning or statistical models can be developed to develop a holistic contextual understanding of the user's current state to help determine whether the present moment is an opportune moment. This contextual understanding could involve attempting to capture the user's feelings, emotions, and prior cognitive workload, as features of the statistical or machine learning models.

10.2 Prototype Updates

10.2.1 Hierarchical Interruptions

The existing prototype interrupts the user of the system by delivering notifications in a timely manner. The individual that is setting up the notification-delivery is typically a mentor or teacher. Naturally, a future iteration of the prototype will enable users aside from mentors and teachers to use the tool to deliver notifications. One possible extension will be the delivery of notifications from a peer in a collaborative project setting, where students are working together towards a common goal of building a software product. This is typically the case when students are in classrooms and working in a group setting. There would need to be functionality to logically group different students together so that only students in the same group would be able to message each other. The default with this solution would be that there would only be

one group with all of the students inside of the group, and all equally able to message each other.

Gupta et al. [19] conducted a series of experiments capturing primary-task performance impact when notifications were delivered, via an instant messaging (IM) system, from different hierarchical relationships, such as peer-to-peer, manager-to-peer, and peer-to-manager. Participants in the research study were interrupted by different hierarchical characters while they were performing predefined search tasks. Ultimately, they concluded that messages sent via IM decrease the overall primary task quality and increase the overall cognitive load; the relationship of the sender and receiver impacts the severity of primary task quality and cognitive load. Interruptions from a supervisor typically increase the time required to complete a task, whereas the opposite effect is present with interruptions from peers. Interestingly, the results here are suggesting that communication between peers typically will result in higher productivity, and resultingly it should be promoted more to improve overall task quality.

10.2.2 MultiShapeCreator Implementation

The prototype is meant to be used in other products built with STaBL.rocks. The primary use case is for the MultiShapeCreator, and other tools in the Teacher Dashboard. That is, the prototype will be integrated into the Teacher Dashboard such that a teacher can send notifications to a group of students while they are using the MultiShapeCreator. The first step in developing the implementation will be to use Hierarchical Task Analysis (HTA) [2] to determine the user goals so that task phases

can be exploited. The HTA for MultiShapeCreator was completed in the Methodology section, identifying a user’s goals and tasks while using MultiShapeCreator. With the results of the HTA, logic can be built by a developer to capture when a user has completed a task or is starting a new task; this is the most opportune time to interrupt a user. As the developer is building the logic to exploit task phases, they can leverage the Types that were used to build the prototype. Instead of developing a new solution, they can import the prototype module into their program, and use the existing functions and types to connect elements of the MultiShapeCreator to the interruption management system. This is possible because both the prototype and MultiShapeCreator follow the same software paradigm of Local-Global Model View Update; both programs have the same structure to track and update the models and subsequently the views. Ultimately, this should lead to a seamless integration between the two programs.

10.2.3 User Interface (UI) Features

Several enhancements to the user interface of the prototype were elicited through the collection of feedback from four participants that are actively engaged in mentoring and instructing students in Elm programming on STaBL.rocks. The enhancements can be grouped into two buckets: (a) user interface changes, (b) role changes. User interface changes comprised of enhancements to the overall look and feel of the prototype to make it a smoother experience for the users. For example, it is suggested that the personified emoji in Figure 5.9 should be made more transparent so that it is less disruptive. Further, the emoji should have either a rising hand, message bubbles, or both to imply the intent of why the emoji is becoming more upset as the bounded

deferral timer is expiring. Role changes enhance the privileges different users can have within the prototype. Administrators can control which students should receive interruptions, and which ones should not. There can also be “enhanced-student” privileges for students who are “team-leads” for a particular session. Custom avatars can be used to identify students and their roles more easily. The aforementioned changes would require a change to the list of shapes displayed for the personified emoji, and/or the local and global models for the prototype. For example, in order to only deliver interruptions to a subset of users, there would need to be changes made to the Teacher Modal to enable the functionality, and the tracking of user ids associated with each interruption in the global model. A change to the personified emoji would only require a View change and not a Model or Update change because only the UI component would be changing for the emoji. The raising hand and jittery or shaking feature would be a function of the remaining time left in the bounded deferral timer, which is precisely how the current process works with the happy, neutral, and sad face.

Chapter 11

Conclusion

The primary contribution of this thesis is in framing a design conversation about interruption management systems (IMSs) for McMaster Start Coding and STaBL Foundation with a literature review and two cycles of prototype and feedback (Chapters 5 & 7) developed in the functional programming language Elm, the language used in teaching and tool development by the target organizations. To make future design discussions more effective, we also propose a methodology to evaluate solutions to the problem (Chapter 3) **“How might we interrupt students for announcements or to collect feedback, such that it results in the least amount of disruption to the interrupted student?”**

Coding mentors are tasked with determining engagement levels of up to 30 students simultaneously, and must interrupt students with the least negative impact on their productivity. Coding mentors can be aided in this task by the IMS prototype and can use the prototype to send announcements to, or collect feedback from students in a virtual or in-person classroom setting. The evaluation methodology is aligned with the Action Research Standards for Design Science, and presents an

approach to capture the impact of the solution while being used by coding mentors.

Inopportune notifications result in detrimental effects on a user's primary task performance [36, 11, 18]. Developing IMSs aids in programmatically determining when to interrupt a user at an opportune moment, so that there is minimal disruption to primary task performance. While the IMS prototype enables coding mentors and solves the defined problem statement, it is limited in scope to the Elm programming language and the STaBL.rocks integrated development environment. Future research should aim to enable coding mentors in different programming languages and classroom settings, such as a hybrid classroom setting, to further understand the effectiveness of IMSs in hybrid classroom settings. The theoretical design of the IMS prototype can also be applied to other existing software tools, to leverage Hierarchical Task Analysis (HTA) to enable an IMS. Furthermore, the theoretical design of the IMS prototype also revealed gaps in the existing literature, leading to opportunities for novel research. The three general areas for novel research are: *(a) developing models to predict task boundaries with programming code as an input feature, (b) large scale empirical studies to identify task boundaries, and (c) developing a methodology to know when to defer notifications to another opportune moment.*

IMSs can be used in virtual or in-person classroom settings to enable coding mentors to support more students than is currently possible. By equipping coding mentors with an IMS, they will have more time and cognitive availability to focus on more technically difficult problems, or support more students in the same task. Students will be able to perform at a higher level too as there are less inopportune interruptions. Ultimately, this research shows that a well-crafted interruption management system can enable students to perform better and enable coding mentors to focus on

more difficult tasks, or complete the same task faster.

Appendix A

Question Classification Text

In this study, we have used a bank of questions developed by Pasupathi et al. [39]. We include it here for completeness.

NO-1 How likely are you to recommend this activity to a friend?

FI-4 All team members, including me, contributed meaningfully.

FI-5 Did you enjoy the planning process of the game: why or why not?

FI-6 The next time you write a [kid creation], will you use the same storyboarding and techniques you learnt in this [camp].

FI-7 Would friends in your class like to create a [Mystery game] together in the same way?

FI-8 Did you enjoy the planning process of the [Mystery game] using the Google Slides we provided?

FA-1 I lost myself in this experience.

FA-2 The time I spent using [Application X] just slipped away.

FA-3 I was absorbed in this experience.

FA-4 When I was doing task, I lost track of the world around me.

PU-1 I felt frustrated while using this [Application X].

PU-2 I found this [Application X] confusing to use.

PU-3 I could not do some of the things I needed to do while using [Application X].

AE-1 This [Application X] was attractive.

AE-2 This [Application X] was aesthetically appealing.

AE-3 This [Application X] was appealed to my senses.

EN-1 using [Application X] was worthwhile.

EN-2 My experience was rewarding.

EN-3 I consider my experience a success.

EN-4 I would recommend this [Application X] to my friends and family

FI-1 I was really drawn into finding the stories.

FI-2 I felt involved in this task.

FI-3 This experience was fun.

NO-1 I continued to use [Application X] out of curiosity

NO-2 I felt interested in the [Application X].

CO-1 I'm pretty happy with who I am.

CO-2 Sometimes I think I am no good at all.

CO-3 I think there are some good things about me.

CO-4 I can do things just as well as other people.

CO-5 I do not have much to be proud of.

CO-6 I feel useless sometimes.

CO-7 I feel like I'm worth something.

CO-8 I wish I liked myself more.

CO-9 I feel like a failure.

CO-10 I think I have a good attitude about myself.

SO-1 I compliment others often.

SO-2 I get along well with others who are different than me.

SO-3 I can clearly describe how I feel.

SO-4 I am respectful of others when they disagree with me.

SO-5 I can stand up for myself without putting others down.

SO-6 I can disagree with someone without starting a fight.

SO-7 I listen carefully to other people's points of view.

SO-8 I care a lot about the feelings of others.

CC-1 I find the things I learn in coding class interesting.

CC-2 I use ideas from coding class often in my daily life.

CC-3 It is important that I do well in coding class.

CC-4 I think coding will be useful for me in the future.

SE-1 I often come to class prepared.

SE-2 I follow directions in class.

SE-3 I do not save my work for the last minute.

SE-4 I can pay attention and ignore distractions.

SE-5 I can easily focus when working by myself.

SE-6 I don't interrupt people.

SE-7 I feel like I can complete all my class assignments.

SE-8 I feel that I can understand complicated ideas.

SE-9 I feel that I can understand complicated ideas.

SE-1 0 I can learn all the topics taught in class.

SE-1 1 I could do the hardest work assigned in class.

SE-1 2 I will remember everything I learned in class next year.

ER-1 It is easy for me to keep my temper under control.

ER-2 I can stay calm even when someone is bothering me.

ER-3 I don't interrupt people.

ER-4 I am polite to adults.

ER-5 I am polite to my peers.

ER-6 I can pull myself out of a bad mood.

ER-7 I can stay relaxed when everyone around me is angry.

ER-8 I can control my emotions when I need to.

ER-9 Once I get upset, it is easy to get myself to relax again.

ER-10 When things go wrong, I can remain calm.

GM-1 People are either born talented or they aren't.

GM-2 People cannot change the school subjects they like to study.

GM-3 You cannot change someone's level of intelligence.

GM-4 Someone cannot change how much effort they can give.

GM-5 You cannot change if you are a misbehaved kid.

GM-6 You cannot change how easily you give up.

Appendix B

Interruption Management System Prototype Types, States, and Messages

B.1 Local Messages

```
type LocalMsg
    = Tick Float GetKeyState
    | OpenTeacherScreen
    | CloseTeacherScreen
    | EnterITChange InterruptionType
    | EnterMsgChange String
    | EnterBDChange String
    | ResetTeacherScreen
    | EnterStudentId String
```

```
| OpenStudentScreen
| CloseStudentScreen
| OpenStudentNotification
| EnterStudentFeedback String
| CloseStudentNotification
```

B.2 Local State

```
type LocalState
  = Start
  | TeacherScreenUsage
  | StudentScreenUsage
  | StudentNotificationScreen
```

B.3 Global Messages

```
type GlobalMsg
  = TeacherPush Interruption
  | StudentPush InterruptionId Response
```

B.4 Type Aliases

Screen Types

```
type InterruptionType = Announcement | Feedback
type User = Student | Teacher
```

Local Model

```
type alias LocalModel =  
  { time : Float,  
    state : LocalState,  
    interruption: Interruption,  
    response: Response,  
    studentId: StudentId  
  }
```

Global Model

```
type alias GlobalModel =  
  { interruptionQueue: InterruptionQueue }
```

Interruption

```
type alias Interruption =  
  { id: InterruptionId,  
    interruptionType: InterruptionType,  
    message: Message,  
    boundedDeferralTime: BoundedDeferralTime,  
    responses: Responses  
  }
```

Response

```
type alias Response =  
  {studentId: StudentId,
```



```
response: String
}
```

Supporting Interruption Type Aliases

```
type alias InterruptionQueue = List Interruption
type alias InterruptionId = Maybe Float
type alias Message = String
type alias BoundedDeferralTime = Float
type alias Responses = List Response
type alias StudentId = Maybe Float
```

Bibliography

- [1] N. Ahmadpour, L. Loke, C. Gray, Y. Cao, C. Macdonald, and R. Hart. Understanding how technology can support social-emotional learning of children: a dyadic trauma-informed participatory design with proxies. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17, Hamburg Germany, Apr. 2023. ACM. ISBN 978-1-4503-9421-5. doi: 10.1145/3544548.3581032. URL <https://dl.acm.org/doi/10.1145/3544548.3581032>.
- [2] J. Annett. Training for Perceptual Skills. *Ergonomics*, 9(6):459–468, Sept. 1966. ISSN 0014-0139. doi: 10.1080/00140136608964410. URL <https://doi.org/10.1080/00140136608964410>. Publisher: Taylor & Francis .eprint: <https://doi.org/10.1080/00140136608964410>.
- [3] K. Arai and R. Bhatia, editors. *Advances in Information and Communication: Proceedings of the 2019 Future of Information and Communication Conference (FICC), Volume 2*, volume 70 of *Lecture Notes in Networks and Systems*. Springer International Publishing, Cham, 2020. ISBN 978-3-030-12384-0 978-3-030-12385-7. doi: 10.1007/978-3-030-12385-7. URL <http://link.springer.com/10.1007/978-3-030-12385-7>.

- [4] M. M. Arias Valencia. Principles, Scope, and Limitations of the Methodological Triangulation. *Investigacion y Educacion en Enfermeria*, 40(2):e03, Sept. 2022. ISSN 0120-5307. doi: 10.17533/udea.iee.v40n2e03. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9714985/>.
- [5] B. P. Bailey, J. A. Konstan, and J. V. Carlis. The Effects of Interruptions on Task Performance, Annoyance, and Anxiety in the User Interface.
- [6] V. Braun, , and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, Jan. 2006. ISSN 1478-0887. doi: 10.1191/1478088706qp063oa. URL <https://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa>. Publisher: Routledge _eprint: <https://www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa>.
- [7] P. Börjesson, W. Barendregt, E. Eriksson, and O. Torgersson. Designing technology for and with developmentally diverse children: a systematic literature review. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 79–88, Boston Massachusetts, June 2015. ACM. ISBN 978-1-4503-3590-4. doi: 10.1145/2771839.2771848. URL <https://dl.acm.org/doi/10.1145/2771839.2771848>.
- [8] N. Carter, D. Bryant-Lukosius, A. DiCenso, J. Blythe, and A. J. Neville. The use of triangulation in qualitative research. *Oncology Nursing Forum*, 41(5):545–547, Sept. 2014. ISSN 1538-0688. doi: 10.1188/14.ONF.545-547.
- [9] E. B. Cutrell, M. Czerwinski, and E. Horvitz. Effects of Instant Messaging Interruptions on Computing Tasks.

- [10] E. C. M. Czerwinski and E. Horvitz. Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance.
- [11] M. Czerwinski, E. Cutrell, and E. Horvitz. Instant Messaging and Interruption: Influence of Task Type on Performance.
- [12] C. d’Alves, T. Bouman, C. Schankula, J. Hogg, L. Noronha, E. Horsman, R. Siddiqui, and C. K. Anand. Using Elm to Introduce Algebraic Thinking to K-8 Students. *Electronic Proceedings in Theoretical Computer Science*, 270:18–36, May 2018. ISSN 2075-2180. doi: 10.4204/EPTCS.270.2. URL <http://arxiv.org/abs/1805.05125v1>.
- [13] R. F. Dam. The 5 Stages in the Design Thinking Process, Mar. 2024. URL <https://www.interaction-design.org/literature/article/5-stages-in-the-design-thinking-process>. [Accessed 2024-10-09].
- [14] N. K. Denzin. Triangulation. In *The Blackwell Encyclopedia of Sociology*. John Wiley & Sons, Ltd, 2015. ISBN 978-1-4051-6551-8. doi: 10.1002/9781405165518.wbeost050.pub2. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405165518.wbeost050.pub2>. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781405165518.wbeost050.pub2>.
- [15] F. A. Dreger, M. Englund, F. Hartsch, T. Wagner, D. Jaeger, R. Björheden, and G. Rinkenauer. Hierarchical Task Analysis (HTA) for Application Research on Operator Work Practices and the Design of Training and Support Systems for Forestry Harvester. *Forests*, 14(2):424, Feb. 2023. ISSN 1999-4907. doi: 10.3390/f14020424. URL <https://www.mdpi.com/1999-4907/14/2/424>. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

- [16] B. Ferreira, M. Kalinowski, M. V. D. C. Gomes, M. C. Marques, H. Lopes, and S. D.J. Barbosa. Investigating Problem Definition and End-User Involvement in Agile Projects that Use Lean Inceptions. In *XX Brazilian Symposium on Software Quality*, pages 1–10, Virtual Event Brazil, Nov. 2021. ACM. ISBN 978-1-4503-9553-3. doi: 10.1145/3493244.3493268. URL <https://dl.acm.org/doi/10.1145/3493244.3493268>.
- [17] K. Frame, O. Neumann, A. Schill, and S. Stocker. An Infrastructure for Collaborative Teletesting.
- [18] T. Gillie and D. Broadbent. What makes interruptions disruptive? A study of length, similarity, and complexity. *Psychological Research*, 50(4):243–250, Apr. 1989. ISSN 0340-0727, 1430-2772. doi: 10.1007/BF00309260. URL <http://link.springer.com/10.1007/BF00309260>.
- [19] A. Gupta, H. Li, and R. Sharda. Should I send this message? Understanding the impact of interruptions, social hierarchy and perceived task complexity on user performance and perceived workload. *Decision Support Systems*, 55(1):135–145, Apr. 2013. ISSN 01679236. doi: 10.1016/j.dss.2012.12.035. URL <https://linkinghub.elsevier.com/retrieve/pii/S0167923613000055>.
- [20] N. Gwangwava. Learning Design Thinking Through a Hands-On Learning Model. *International Journal of Innovative Teaching and Learning in Higher Education*, 2(1):1–19, Apr. 2021. ISSN 2644-1624, 2644-1640. doi: 10.4018/IJITLHE.20210101.0a4. URL <https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/IJITLHE.20210101.0a4>.

- [21] E. Horvitz, C. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communication: from principles to applications. *Communications of the ACM*, 46(3):52–59, Mar. 2003. ISSN 0001-0782, 1557-7317. doi: 10.1145/636772.636798. URL <https://dl.acm.org/doi/10.1145/636772.636798>.
- [22] S. T. Iqbal, P. D. Adamczyk, X. S. Zheng, and B. P. Bailey. Towards an index of opportunity: understanding changes in mental workload during task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 311–320, Portland Oregon USA, Apr. 2005. ACM. ISBN 978-1-58113-998-3. doi: 10.1145/1054972.1055016. URL <https://dl.acm.org/doi/10.1145/1054972.1055016>.
- [23] A. J. Ko, B. A. Myers, and H. H. Aung. Six Learning Barriers in End-User Programming Systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 199–206, Rome, Italy, 2004. IEEE. ISBN 978-0-7803-8696-9. doi: 10.1109/VLHCC.2004.47. URL <https://ieeexplore.ieee.org/document/1372321/>.
- [24] S. Kurkovsky. Four roles of instructor in software engineering projects. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 354–354, Madrid Spain, June 2008. ACM. ISBN 978-1-60558-078-4. doi: 10.1145/1384271.1384396. URL <https://dl.acm.org/doi/10.1145/1384271.1384396>.
- [25] K. A. Latorella. Effects of Modality on Interrupted Flight Deck Performance: Implications for Data Link. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 42(1):87–91, Oct. 1998. ISSN 1071-1813, 2169-5067.

- doi: 10.1177/154193129804200120. URL <https://journals.sagepub.com/doi/10.1177/154193129804200120>.
- [26] M. J. Lee and A. J. Ko. Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the seventh international workshop on Computing education research*, pages 109–116, Providence Rhode Island USA, Aug. 2011. ACM. ISBN 978-1-4503-0829-8. doi: 10.1145/2016911.2016934. URL <https://dl.acm.org/doi/10.1145/2016911.2016934>.
- [27] S. W. Lee and D. C. Rine. Missing requirements and relationship discovery through proxy viewpoints model. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1513–1518, Nicosia Cyprus, Mar. 2004. ACM. ISBN 978-1-58113-812-2. doi: 10.1145/967900.968203. URL <https://dl.acm.org/doi/10.1145/967900.968203>.
- [28] K. Lewin. Untersuchungen zur Handlungs- und Affektpsychologie.
- [29] J. Liedtka. Perspective: Linking Design Thinking with Innovation Outcomes through Cognitive Bias Reduction. *Journal of Product Innovation Management*, 32(6):925–938, Nov. 2015. ISSN 0737-6782, 1540-5885. doi: 10.1111/jpim.12163. URL <https://onlinelibrary.wiley.com/doi/10.1111/jpim.12163>.
- [30] D. S. McCrickard and C. M. Chewar. Attuning notification design to user goals and attention costs. *Communications of the ACM*, 46(3):67–72, Mar. 2003. ISSN 0001-0782, 1557-7317. doi: 10.1145/636772.636800. URL <https://dl.acm.org/doi/10.1145/636772.636800>.
- [31] D. S. McCrickard, C. M. Chewar, J. P. Somervell, and A. Ndiwalana. A

- model for notification systems evaluation—assessing user goals for multitasking activity. *ACM Transactions on Computer-Human Interaction*, 10(4):312–338, Dec. 2003. ISSN 1073-0516, 1557-7325. doi: 10.1145/966930.966933. URL <https://dl.acm.org/doi/10.1145/966930.966933>.
- [32] D. McFarlane and K. Latorella. The Scope and Importance of Human Interruption in Human-Computer Interaction Design. *Human-computer Interaction*, 17: 1–61, Mar. 2002. doi: 10.1207/S15327051HCI1701_1.
- [33] D. C. McFarlane. Interruption of People in Human-Computer Interaction: A General Unifying Definition of Human Interruption and Taxonomy:. Technical report, Defense Technical Information Center, Fort Belvoir, VA, Dec. 1997. URL <http://www.dtic.mil/docs/citations/ADA333587>.
- [34] A. Mehrotra and M. Musolesi. Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges, Jan. 2018. URL <http://arxiv.org/abs/1711.10171>. arXiv:1711.10171 [cs].
- [35] Y. Miyata and D. Norman. *Psychological issues in support of multiple activities*. Jan. 1986.
- [36] Y. Miyata and D. A. Norman. Psychological Issues in Support of Multiple Activities. In *User Centered System Design*, pages 265–284. CRC Press, Boca Raton, 0 edition, Jan. 1986. ISBN 978-1-4822-2963-9. doi: 10.1201/b15703-13. URL <https://www.taylorfrancis.com/books/9781482229639/chapters/10.1201/b15703-13>.

- [37] I. L. Nunes, editor. *Advances in Human Factors and System Interactions: Proceedings of the AHFE 2016 International Conference on Human Factors and System Interactions, July 27-31, 2016, Walt Disney World®, Florida, USA*, volume 497 of *Advances in Intelligent Systems and Computing*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-41955-8 978-3-319-41956-5. doi: 10.1007/978-3-319-41956-5. URL <http://link.springer.com/10.1007/978-3-319-41956-5>.
- [38] P. Pasupathi. Sd draw: A state diagram tool including elm code generation for interactive applications. Master’s thesis, McMaster University, 2021.
- [39] P. Pasupathi, C. W. Schankula, N. DiVincenzo, S. Coker, and C. K. Anand. Teaching Interaction using State Diagrams, July 2022. URL <http://arxiv.org/abs/2207.12701>. arXiv:2207.12701.
- [40] M. Q. Patton. Enhancing the quality and credibility of qualitative analysis. *Health Services Research*, 34(5 Pt 2):1189–1208, Dec. 1999. ISSN 0017-9124.
- [41] N. Peters, B. Raj, and G. Romigh. Topic and Prosodic Modeling for Interruption Management in Multi-User Multitasking Communication Interactions.
- [42] N. S. Peters. *Collaborative Communication Interruption Management System (C-CIMS): Modeling Interruption Timings via Prosodic and Topic Modeling for Human-Machine Teams*. Ph.D., Carnegie Mellon University, United States – Pennsylvania, 2017. URL <https://www.proquest.com/docview/2006928903/abstract/E42057D9F3AB4E90PQ/1>. ISBN: 9780355539332.
- [43] P. Ralph, N. b. Ali, S. Baltes, D. Bianculli, J. Diaz, Y. Dittrich, N. Ernst,

- M. Felderer, R. Feldt, A. Filieri, B. B. N. d. França, C. A. Furia, G. Gay, N. Gold, D. Graziotin, P. He, R. Hoda, N. Juristo, B. Kitchenham, V. Lenarduzzi, J. Martínez, J. Melegati, D. Mendez, T. Menzies, J. Moller, D. Pfahl, R. Robbes, D. Russo, N. Saarimäki, F. Sarro, D. Taibi, J. Siegmund, D. Spinellis, M. Staron, K. Stol, M.-A. Storey, D. Taibi, D. Tamburri, M. Torchiano, C. Treude, B. Turhan, X. Wang, and S. Vegas. Empirical Standards for Software Engineering Research, Mar. 2021. URL <http://arxiv.org/abs/2010.03525>. arXiv:2010.03525 [cs].
- [44] Rhode Island School of Design and M. Leighton Beaman. By Proxy: Design Problems and Collaborative Inquiry. In *Intersections Between the Academy and Practice, Papers from the 2017 AIA/ACSA Intersections Symposium*, pages 15–21. ACSA Press, 2017. ISBN 978-1-944214-13-5. doi: 10.35483/ACSA.AIA.Inter.17.4. URL <https://www.acsa-arch.org/chapter/by-proxydesign-problems-and-collaborative-inquiry/>.
- [45] P. G. Rowe. *Design Thinking*. MIT Press, Feb. 1991. ISBN 978-0-262-68067-7. Google-Books-ID: ZjZ3mflzJtUC.
- [46] C. Schankula, S. Smith, and C. Anand. *TEASync: A Novel Framework for Approachable, Serverless Concurrent Web Applications*. Apr. 2024.
- [47] A. Shivakumar, A. Bositty, N. S. Peters, and Y. Pei. Real-Time Interruption Management System for Efficient Distributed Collaboration in Multi-tasking Environments. *Proceedings of the ACM on Human-Computer Interaction*, 4 (CSCW1):1–23, May 2020. ISSN 2573-0142. doi: 10.1145/3392844. URL <https://dl.acm.org/doi/10.1145/3392844>.

- [48] D. Siemon, F. Becker, and S. Robra-Bissantz. How Might We? From Design Challenges to Business Innovation. 4:96–110, Dec. 2018.
- [49] J. H. Walker, L. Sproull, and R. Subramani. Using a human face in an interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 85–91, New York, NY, USA, Apr. 1994. Association for Computing Machinery. ISBN 978-0-89791-650-9. doi: 10.1145/191666.191708. URL <https://dl.acm.org/doi/10.1145/191666.191708>.
- [50] S. Willermark, A. S. Islind, T. Appelgren, and M. E. Taavo. The polite pop-up: An experimental study of pop-up design characteristics and user experience.
- [51] J. Zhang, A. Verma, C. Sheth, C. W. Schankula, S. Koehl, A. Kelly, Y. Irfan, and C. K. Anand. Graphics Programming in Elm Develops Math Knowledge & Social Cohesion. 2018.