CREATING ACCESSIBLE UML CLASS DIAGRAMS

By

Ira Woodring

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2025

ABSTRACT

Unified Modeling Language (UML) Class Diagramming is the commonly accepted mechanism used to describe relationships between software components. In addition, it is an essential educational tool that is used to convey the structure of software and the patterns of software design to students. Unfortunately, UML is a visual-only mechanism and therefore is not useful for developers and students who are blind or have visual impairments. This work describes a method for conveying class diagrams using audio, which addresses this lack of a tool to support these populations. This method works by dividing the views of a diagram into smaller spaces. Elements in these subspaces are conveyed through manipulation of audio properties. Multiple user studies were performed to prove that the tool is viable for conveying the static structure of software elements and that the workload required to use the tool is reasonable. The results of the studies indicate that the tool is effective and requires only slightly higher mental workload than traditional class diagrams.

This thesis is dedicated to my wife, Sarah, who believed in me, to Sue and Eric Dravland who showed me I was worth loving, and to my kids who gave me the strength to keep going.

ACKNOWLEDGEMENTS

Thank you to Michael Hudson for your contributions and support. Your perspective has been eye-opening and invaluable.

Thank you to my colleagues at Grand Valley State University. Without your support, friendly ears, and advice, I would not be where I am today.

Most of all, thank you, Dr. Charles Owen for your patience, wisdom, and understanding. Your guidance and support made this work possible. I have had few great teachers over the years, and even fewer who show the compassion, care, and dedication that you provide to your students. Thank you, Sir.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

UML Class Diagrams are the most commonly accepted method of modeling software. However, presently there is a glaring inadequacy in that they are only usable for developers without visual impairments or blindness. Although experienced developers may overcome this inadequacy, computing students with disabilities are left to struggle while learning complex computing topics without these vital aids.

According to the US National Center for Education Statistics, 14% of American schoolchildren (7.3 million total children) qualify as having a disability under the Individuals with Disabilities Education Act (IDEA) [7]. As these students matriculate into the workforce and higher education, it is imperative that they receive as good an education as their peers. For students who major in software engineering or related fields, the inability to use a tool as powerful and prevalent as class diagrams makes this impossible.

Furthermore, in the last decade, multiple countries have accepted that at least a cursory understanding of computer science concepts is essential for their citizens. The United Kingdom, for example, has added computing competencies to its national curriculum, and President Obama created AccessCSforall [1] in the United States [5, 4]. Obama noted that computer science should become a "basic skill" for the citizenry. Implicit in this endeavor is the mandate to ensure equal access to computing education for students with disabilities.

At the same time, the complexity of many modern software solutions has become immense. For example, the F35 Joint Strike Force jet in the United States has an estimated 24 million lines of code [65]. The code base for Google services is estimated to be more than 2 billion lines of code [54]. Even small desktop applications often have a large number of lines of code; the popular open-source VLC media player, for example, consists of over 32 million lines[2]. Table 1.1 presents the number of lines of code for several popular open-source projects.

[1] https://www.washington.edu/accesscomputing/accesscsforall

[2] As measured by the Github-Linguist tool available at https://github.com/github/linguist

Object-oriented design of software projects allows for dividing code into areas of concern and thus managing this complexity. Alan Kay, the creator of Smalltalk (the first object-oriented programming language), noted that "Object-oriented design is a successful attempt to qualitatively improve the efficiency of modeling the ever more complex dynamic systems and user relationships made possible by the silicon explosion" [39]. However, even when divided into areas of concern, there needs to be a way to easily represent the interrelationships between various object types. The computing field makes use of many different diagram types, for various purposes, such as Class Responsibility Collaboration (CRC) cards, Entity Relationship (ER) diagrams, and Unified Modeling Language (UML) diagrams. Traditionally, static software structure has been conveyed through UML class diagrams. However, to include all individuals, a different method is necessary.

Project	**Lines of Code**
Linux Kernel	1,010,549,062
Swift	92,988,974
Rust	53,603,913
VScode	42,293,424
VLC Media Player	32,920,033
React	6,143,496
Vuejs	2,926,351

Table 1.1 Lines of code (as measured by Github-Linguist) of selected popular open-source projects. Represents the state of projects from March 2022.

UML is a visual language for modeling systems [16]. It consists of multiple diagram types that represent systems and interactions with and within systems. Of the various types of UML diagrams, UML Class Diagrams are used for modeling relationships between code entities in object-oriented software projects. Many undergraduate computer science programs teach class diagrams, and it is generally considered the standard way to model software. Although some studies show that UML use is decreasing ([12]), it is still a valuable tool for representing the relationships between components in a software design and a powerful pedagogical tool.

Previous research demonstrates that properly designed sonification and auditory displays

enhance data presentation and comprehension across various domains. For example, studies have shown that sonifying image features help listeners better understand and reproduce image details through drawing [78]. In medical settings, sonification has been shown to be as effective as visual presentation in determining respiratory information, even in busy environments [72]. Individuals monitoring computer network traffic experienced reduced workload and improved awareness of traffic types and attacks when using auditory displays [24]. The offer of multiple modalities for data interaction improves user experience and understanding.

1.0.1 UML Class Diagrams

Unified Modeling Language (UML) is a visual language to describe aspects of software systems. Class diagrams provide UML with the ability to model the static structure of a system [16]. They provide developers and maintainers of systems with an understanding of the overall architecture, as well as information about the relationships between a system's elements. Class diagrams are a type of node-edge graph, where nodes represent software elements, and edges represent the relationships between them.

However, UML is not a perfect solution. Students often struggle to learn and use UML diagrams [55, 22, 60]. The problems of learning UML arise due to the complexity of the language, as well as inadequate materials for teaching and learning UML [60]. However, the most notable inadequacy is that UML is a visual language and therefore not useful to students who are blind or have visual impairments. Unfortunately, previous research has concluded that it is impossible to fully include these individuals without having the assistance of an individual without visual impairment due to the lack of available and appropriate tools [42].

Using supplemental or alternate representations of information can benefit educators and students in this regard. By providing supplemental data representations, students receive an additional mechanism for understanding and exploration. The provision of alternate representations ensures that more students have access and can overcome some of the barriers imposed by differing abilities and schemas. For example, in a 2021 study, students were

taught the bubble, insert, selection, and merge sorting algorithms in short 15-minute lessons with the help of audio[7]. During the lessons, participants were instructed to run a program that used the algorithm they had just learned to sort 2500 random numbers. Participants were exposed to an output of the time taken, an audio sonification of the run as a sorting visualization, or a combined sonification and visualization. Two weeks after the lessons were completed, the students received a survey quiz to assess learning and retention. The results showed that the participants who received the sonification performed as well as the students who received graphical representations. Similarly, researchers have had success with multiple mode methods of teaching arrays and other data structures[50, 18]. Researchers have suggested that a computer science curriculum that includes multiple modalities results in a significant increase in reports of programming self-efficacy in blind students [61].

It has also been shown that users often prefer multi-modal methods of interfacing with data. In their ChartMaster work, which seeks to present Stock Market data effectively with screen readers, Zou and Treviranus noted that users wanted options and that no single modality for presenting the information (among speech, sonification, haptic and hybrid methods) was preferred by all users [80]. Doush et al. reinforce this in their work by displaying charts from common office productivity software; their guidelines for graphical representations specifically call for presentation strategies that can be customized and affected by user control [10].

UML provides for a variety of diagram types, each with its own node and edge types. The nodes in class diagrams usually indicate a class structure, although they are generic enough to be used for interfaces, enumerations, and other types of entities. One of the most important element types of a class diagram are the edges between the nodes, which represent the relationship types (Figure 1.1).

Relationships between classes in object-oriented code bases are one of four types. We list them here in alphabetical order. The first type, “associations”, indicates that one class holds a reference to one or more instances of another class as an instance variable. There

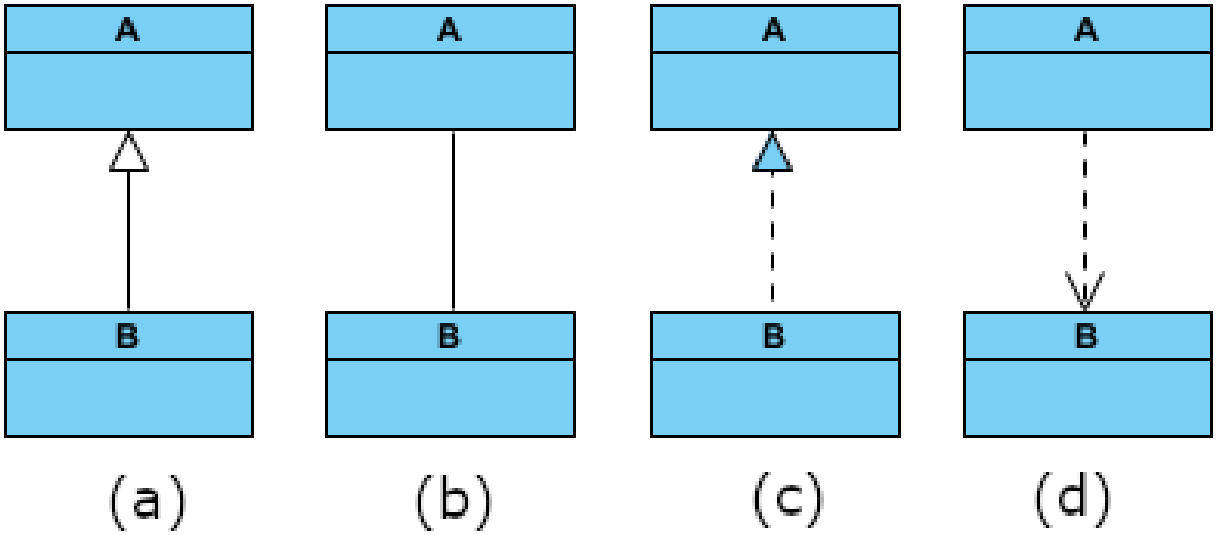

Figure 1.1 Elements of a class diagram include classes (represented as blue rectangles), and a variety of arrows to represent different relationship types between them. (a) A generalization relationship. (b) An association relationship. (c) A realization relationship. (d) A dependency relationship.

are two additional special types of associations, "aggregations" and "compositions". We do not go into those here, as a generic association type is adequate for our work.

A "dependency" exists where a class relies upon the existence of another but does not inherit from that class and does not hold an instance variable of the other class's type. These may exist because the first class has a local variable of the second type, takes the second type as a parameter to one of its methods, or returns the second type from one of its methods.

"Generalizations" indicate that a class inherits information from another class. Inheritance is vital to object-oriented programming, allowing for simplicity in the creation of polymorphic elements. "Realizations" indicate the use of an interface. Not all computer languages support interfaces, and it is possible to represent this type of relationship with generalizations.

1.0.1.1 Representations of Node-Edge Graphs

Node-edge graphs are found in many varying areas of science and mathematics. Class diagrams are a specialized form of a directed node-edge graph. Node-Edge graphs may be

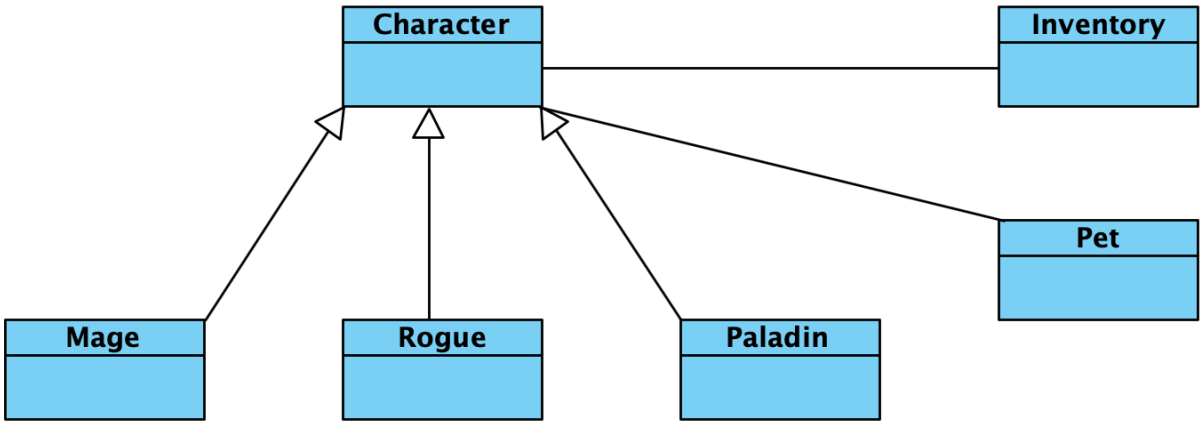

Figure 1.2 A simple class diagram composed of six classes, and the generalization and association relationships between them.

represented in a variety of alternative formats. Consider the diagram in Figure 1.2:

Using set notation, we can mathematically represent the same diagram as a set of nodes and edges between them. Here, we call the diagram G, the nodes V, and the edges E. We can then define the diagram as follows.

$$G = (V, E) \tag{1.1}$$

where

$$V = \{Character, Inventory, Mage, Paladin, Pet, Rogue\} \tag{1.2}$$

$$E = \{(Character, Inventory), (Character, Pet), (Mage, Character), (Rogue, Character), (Paladin, Character)\} \tag{1.3}$$

An adjacency list could also be used to represent the graph. An adjacency list is a list of lists, with each node in the graph containing a list of nodes to which the node connects. Adjacency-list representations work best for graphs that have a smaller number of connections between elements. Our diagram above would be represented as

	Character	Inventory	Mage	Paladin	Pet	Rogue
Character		A			A	
Inventory						
Mage	G					
Paladin	G					
Pet						
Rogue	G					

Table 1.2 An adjacency matrix representation of the diagram from Figure 1.2. Here, we have used the letters "A" and "G" to indicate the connection is an association or generalization.

$$\begin{aligned} &Character - Inventory, Pet \\ &Mage - Character \\ &Rogue - Character \\ &Paladin - Character \end{aligned} \tag{1.4}$$

An adjacency matrix view provides a representation of the same graph in table form. For a graph with n nodes, an adjacency matrix will be an n^2 dimension table.

1.0.2 Rights of Individuals with Disabilities

Judicial precedent (Case or Common Law) and statutes (Civil Law) govern the rights of individuals with disabilities in the workforce, education, and other areas of life. What follows is a summary of some of the more important rules and regulations.

Although it specifically addressed race, the landmark *Brown v. Board of Education of Topeka* [1] is regarded as an important milestone for disability advocates, as it was in many ways the beginning of judicial support for minority rights [45]. Furthermore, as Meyer and Boutcher argue, the strategies used by activists and legal teams for *Brown* were adopted by those working on disability rights initiatives; therefore, its importance for the cause of equity for people with disabilities cannot be overstated.

Pennsylvania Association for Retarded Children (PARC) v. Commonwealth of Pennsylvania (1971) was brought by parents of children with mental disabilities, who argued that

their children were entitled to appropriate state-provided education and services. Furthermore, parents wanted their children to be served in an appropriate age-appropriate school attended by non-handicapped children [3]. The court set an important precedent by finding in the parents' favor, and later legislation mandated this educational model. *Mills v. Board of Education* (1972) reinforced the PARC result. Judge Joseph Cornelius Waddy, who presided over the case, wrote in his decision that public education is "a right which must be made available to all on equal terms."

In 1973, the U.S. Congress passed the *Rehabilitation Act* to prevent discrimination against individuals with disabilities. Section 504 of this law mandates that any entity that receives federal funding must not discriminate solely based on an individual having a disability. The section specifically mentions that it pertains to organizations "principally engaged in the business of providing education, health care, housing, social services, or parks and recreation". Section 508 specifies that electronic and information technology systems should be accessible to persons with disabilities [3].

The *Education for All Handicapped Children Act of 1975* enshrined a federal right for students with disabilities to receive a fair and appropriate public education (FAPE) to take place in the least restrictive environment (LRE) possible depending on a child's needs. In addition to education, the legislation entitled those students to related services to support their education. An important aspect of this law was the requirement that students with disabilities have an individualized education program (IEP) tailored to their specific needs. The law allowed parents to be involved in creating and reviewing the IEP and required states to have procedures on which parents could rely to challenge the IEP. The law was reauthorized in 1990 at which time it was renamed the *Individuals with Disabilities Education Act* (IDEA) and again reauthorized in 2004 to align with *No Child Left Behind* (NCLB) a law that tied educational funding to student achievement on state-mandated performance metrics. As *NCLB* did not add new protections for students with disabilities (and has since

[3]Unless by doing so the agency is imposed with an undue burden.

been mostly replaced), it is not covered in further detail here.

The 1982 case *Board of Education v. Rowley* provided the Supreme Court with its first opportunity to rule on the Education of the Handicapped Act. This suit was brought about by the parents of a child who was deaf. Although their daughter had been somewhat successful academically, they felt that she was not meeting her potential, as she was unable to understand all the words in the classroom. The court sided with the school board, finding that the 1975 act did not intend for students to meet their full potential, but rather to provide them access to a free, appropriate public education.

The *Americans with Disabilities Act* (ADA) of 1990 added additional protections for individuals with both physical and mental disabilities, such as the requirement that businesses provide "reasonable accommodations" to employees with disabilities, requirements for accessible public transportation, accessible design requirements for new and existing public buildings, and other miscellaneous provisions.

1.0.2.1 Implications of Disability Laws

The spirit of the law concerning Americans with disabilities is clear in that alternative representations of information should be made available to students and employees in cases where it does not provide an undue burden on an organization to do so. The effects of these laws exist all around us; for instance, in Braille signage in public buildings and closed captioning in television broadcasts.

However, there exists an obvious lack of accessibility in diagramming. This is, without a doubt, due to the complexities of creating alternative representations of information that meet these specialized use cases. However, as the ability to work with data becomes ever more important across many occupations, the lack of viable alternative representations of data becomes not just inconvenient but economically limiting to the millions of individuals to those who experience it.

1.0.3 W3C Best Practices for Accessibility

With the ubiquitous presence of Web browsers on virtually all commodity computers, as well as the widespread adoption of the Web as the primary platform for sharing content, there has been an increasing focus on the best practices for assistive computing in this domain. Although the W3C[4] exists to develop accessibility standards specifically for the Web, its recommendations serve as one of the most important repositories of best practices to ensure that content is accessible for all people. What follows is a summary of the key points of their Content Accessibility Guidelines [2].

The W3C recommends that content adhere to four principles of accessibility. The first is that the content must be presented in a way that is perceivable by users. In practice, this may mean that there are text-based alternatives to audio-based information such as captions, that there are audio-based descriptions of visual media, or that some other alternative representation(s) of data exist. Furthermore, the content should be adaptable, which means that the presentation of information should be changeable without losing access to the information. Orientation should not be fixed and information should be easily distinguishable. Color schemes in visualizations, and audio streams should provide easily differentiable stimuli.

Second, the user interface must be operable. The W3C recommends that all features of an interface be available from the keyboard and note that careful design should ensure that keyboard focus is never "trapped" in a content area. At the same time, they recommend that alternative input modalities, such as touch screens and speech recognition, be added when possible. Input should not be time-limited, and the timing of any stimuli should be controllable by the user. Furthermore, there should never be more than three flashes in the time span of one second, so as to not trigger seizures.

Third, the information and operation of the user interface should be understandable. Idioms and jargon should be kept to a minimum. The operation of the interface should be

[4]https://www.w3.org/about/

predictable and the navigation consistent throughout the system. Error prevention mechanisms should be prevalent and work well. Changing the focus of a particular structural element should not modify contextual areas such as the view port size or layout.

Lastly, the content must be robust. Here, W3C means that content should be compatible with a large number of technologies, such as screen readers. Changes in user input elements should always provide the user with change notifications. Status messages must be available to assistive technologies.

1.0.4 Interfacing With Screen Readers

The most widely used digital tool for people with blindness or vision impairments is the screen reader. A 2021 survey by the Institute for Disability Research, Policy and Practice found that of 1,568 respondents, 98.7% reported using a screen reader. Of these survey respondents, 53.7% reported the primary use of Job Access With Speech (JAWS) [5] and 30.7% used non-visual desktop access (NVDA) [6]. The Accessibility Principles of the Worldwide Web Consortium (W3C) call for all non-textual content to provide a textual representation [6]. Non-textual content refers to images, buttons, and other controls, as well as multimedia and data represented with charts or other diagrams.

Unfortunately, research has shown that these standards are often ignored. One study determined that only 39.6% of the images on the top 500 high-traffic websites had applied textual representations [15]. This is particularly problematic when an image contains a visualization of data or information. In one study of the frustrations encountered by more than 100 screen readers, the researchers found that lack of appropriate alt-text, poorly labeled or unlabeled forms, poor navigation, and other problems caused considerable frustration and up to 30% lost time when using computers [40].

An analysis by Sharif et al. classified the problems that screen readers faced while browsing websites randomly drawn from a curated set of 50 sites that included visualizations [58]. The problem categories they discovered were

[5] https://www.freedomscientific.com/products/software/jaws/
[6] https://www.nvaccess.org/download/

1. **Invisible Visualizations** - Screen readers did not detect a visualization on 33% of the pages that had them.
2. **Incomprehensible Visualizations** - Visualizations were detected by the screen reader, but were labeled only as "blank", "graphic", etc.
3. **Lack of Access to Any Data Points** - Screen readers could detect a visualization but could not access the data within.
4. **Lack of Holistic Exploration and Trend Assessment** - Users were unable to gain a high-level understanding of the visualizations and data within.
5. **Lack of Ability to Investigate Specific Data Points** - Some data points may be available, but not all.
6. **Lack of Tabular Representation** - Information was only provided via the graphical modality.
7. **Lack of Textual Representation** - Lack of alt-text and other descriptive text.
8. **Lack of Description of Overall Trend in a Non-Visual Format** - No summary information or description of the visualization was provided.

The study designers recommended the appropriate use of alternate text and ARIA attributes, as well as a design that allows for holistic and lower-level exploration, autogenerated alternate text, and multiple modes for exploring data such as sonification.

1.0.5 Contributions of this Thesis

This work and future work based on this work is an effort to make UML class diagrams more accessible to students who are blind or visually impaired while also determining best practices for tools that allow for multimodal exploration of UML class diagrams (and possibly other). Although UML encompasses more types of diagram than just class diagrams, a survey by Muller found that class diagrams are the most widely used in presentations to computer science and software engineering students [46].

Specifically, the efforts presented here were to develop a mechanism for presentation of class diagrams that:

- Does not require manual re-creation of existing diagrams.
- Reproduces diagrams from an existing, widely-used, production quality tool.
- Uses inexpensive, ubiquitous, or nearly ubiquitous technologies for rendering diagrams.

- May require some training, but must not require a great deal of training to use effectively.

The mechanism in this work, the use of sound for supplementing and replacing visualizations, gives presenters a way to overcome the limitations of visualizations. It does this by careful manipulation of psychoacoustic properties, as determined by the authors and other researchers' work.

1.1 Human Perception

The processes of human perception are complicated and not well understood. They comprise much more than the simple reception of physical stimuli. The following work details the knowledge that has informed this work.

1.1.1 Human Visual Perception and Visualization Best Practices

The work in this thesis focuses on human audio perception, but to disregard the well-established knowledge of the human visual system would be folly. Furthermore, the science of human audio perception is not as mature as visual perception; therefore, we look to more mature research to find hints as to how this complicated phenomenon works. Undoubtedly, the physical processes of recognizing waves of light as a picture of a loved one differ from those that recognize waves of audio as a recording of their voice. However, there are important parallels between the psychological processes that perform both tasks. For example, Albert Bregman noted in his work on audio perception that sensory inputs, be they from drawings or recordings, are separated into groups by the nervous system, while it makes sense of patterns in the world [14].

Although classical theories on the evolution of the physical aspects of human vision hypothesized that it developed largely to consume the maximum energy input from the Sun, recent work has indicated that it is likely to have developed to maximize the input of information [25]. However, not all information is useful; thus, human perceptual systems make use of the mechanism of attention to determine which groups of stimuli should require further processing [33]. But what criterion causes the visual perceptual system to form a stimuli group? For instance, what causes human brains to associate the varying waveforms of light detected by our physical processes to be grouped into an entity we could psychologically recognize as an apple? Research shows that objects in a scene are identified by properties such as color, texture, orientation, spatial frequency, brightness, and movement direction, stimuli that are easy for our systems to "separate" from competing stimuli[67].

However, separable stimuli alone are not enough. For example, studies have shown that

the use of disparate colors alone is not enough to create an effective visualization; instead, visualization designers should consider the meaning of colors in context[66]. This implies that the creation of an effective visualization requires a strong understanding not just of the data being presented but of the viewers' mental models and schemas about the data. This is probably why even though there is a great body of visualization research, there is a lack of centralized, holistic guidelines for visualizations[26].

1.1.1.1 Visualization Best Practices

Ben Shneiderman introduces a taxonomy for organizing and understanding the various approaches to information visualization in his often cited work, "The Eyes Have It: A Task by Data Type Taxonomy for Data Visualization" [59]. The central framework of the paper is built around the Visual Information-Seeking Mantra: "Overview first, zoom, and filter, then details on demand." This mantra emphasizes the importance of first giving users an overview of large data sets, allowing them to zoom in and filter data based on their interests, and finally providing detailed information when requested. This process helps to manage the challenge of navigating large and complex data sets efficiently.

Shneiderman proposes a task-by-data-type taxonomy to classify visualization techniques. He identifies seven key data types, including one-dimensional data (like text), two-dimensional data (such as maps), three-dimensional data (real-world objects), temporal data (time-based sequences), multidimensional data (attributes with multiple variables), tree structures (hierarchies), and network data (interconnected items). The paper also outlines seven core tasks that users perform when interacting with visualized data: overview, zoom, filter, details-on-demand, relate, history, and extract. These tasks help users interact meaningfully with the data, from gaining a broad understanding to finding specific details.

The paper further discusses the value of dynamic queries and filtering techniques that allow users to rapidly explore large data sets. By offering smooth user-controlled interactions, these systems facilitate easier navigation and understanding of complex information. Shneiderman also explores the application of various visualization tools, such as treemaps

and fisheye views, and emphasizes the role of advanced interface designs in reducing information overload. Ultimately, the paper argues that effective visualizations must make full use of human perceptual abilities and create intuitive interfaces for a broad range of users.

Although this work focuses on visualizations, the process it identifies is one that translates well to other sensory inputs, as it is really a guide for how to present *information* in any form.

A study by Doush et al. provided ten recommendations for the presentation of graphical information [11]. Although its work focuses on the multimodal presentation of more generic graphical information, some of their guidelines should be considered in the design of a multimodal UML framework. In particular, their work notes the need for:

1. Summary information
2. Customizable presentation strategies
3. User ability to change presentation modality
4. Structural hierarchy views for information for ease of navigation (i.e. chart titles and axis information)
5. Contextual information like pauses and other audio cues
6. Careful choice of scale

These best practices reinforce Shneiderman's results and go further; providing summary information is analogous to providing an overview, for instance. Zooming and filtering allow users to change the presentation modality. Providing contextual information, such as pauses, becomes increasingly important when switching from visual to audio stimuli as the primary method of representing information.

1.1.2 Human Audio Perception

The phenomenon of human audio perception is complex and involves many stages of processing in multiple parts of the body. He noted that humans can perceive light with wavelengths between 380 and 750 nm, resulting in a range of roughly one octave, while

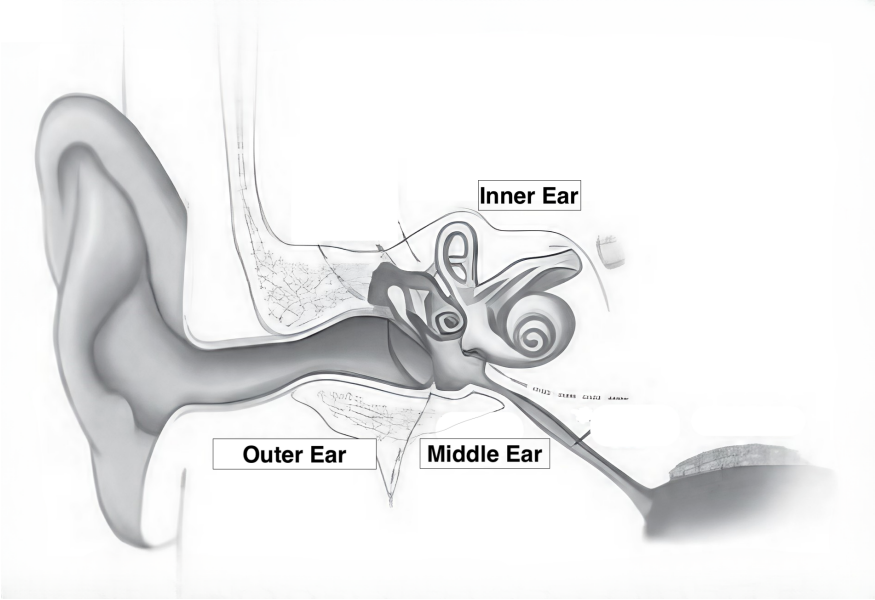

Figure 1.3 The three areas of the ear. Adapted from [63].

humans perceive sound from 20Hz to 20,000Hz, or nearly 10 octaves [49]. The science of audio perception involves biological and psychological processes.

1.1.2.1 Biological Processes of Hearing

The US National Institutes of Health summarizes the basic biological processes of hearing [63]. Vibrations traveling across some media (usually air, but not always) are first picked up by the outer ear. The outer ear is made up of the fleshy portion called the pinna and the ear canal. Interestingly, the physical properties of the pinnae, such as shape, affect the incoming vibrations, resulting in perceptual differences [35].

After the sound travels through the ear canal, frequencies 3,000 - 4,000 Hz are amplified before the sound is received by the tympanic membrane (what many people refer to as the eardrum) [63]. This membrane connects to the ossicles, a group of three small bones in the middle ear. The sound then travels to the inner ear.

For hearing, the most important part of the inner ear is the cochlea. Within the cochlea, vibrations are converted to electrical signals that the brain can consume by means of tiny hair cells called stereocilia. Figure 1.3 shows the three parts of the ear.

1.1.2.2 Psychological Processes of Hearing

Although the biological processes of hearing are complex, the basic physical mechanisms are fairly well understood. In contrast, scientists continue to unravel the mysteries of the psychological side of human hearing, a field dubbed psychoacoustics. Early works by Colin Cherry identified one of the first foundational problems of psychoacoustics, the famous "Cocktail Party" problem[21]. Cherry noted that under certain circumstances, humans can perceive more than one speaker at a time; for example, while we are listening to one person at a cocktail party, we are often able to recognize the speech of another person simultaneously.

Cherry performed tests where subjects listened with one or both ears to auditory stimuli, while he varied the properties of the stimuli, and found that changes in delay caused perceptual effects in the listeners. Given a delay between the two stimuli, participants perceived the listener to move to their left or right side. More interestingly, with a delay of less than 20 ms given the same speech in both ears, the participants perceived the two stimuli as coming from a single source. These results helped researchers realize that sound perception is more than just the sum of physical stimuli.

Later research by Albert Bregman continued to examine these perceptual oddities. Bregman created the term "Auditory Scene Analysis" to describe his theories of human audio perception [14]. Just as the visual system uses separable properties of visual stimuli, the brain relies on separable audio properties to determine which audio stimuli should be grouped. Bregman referred to the groupings as "streams". He was very clear to stipulate that a stream differed from a sound; he noted that a stream is a single event and that streams could be composed of multiple sounds (for instance, he noted that a musical performance might consist of a singer with a piano backing).

Bregman's work often found auditory parallels to existing Gestalt principles of visual perception. These principles - proximity, similarity, uniform density, common fate, closure, direction, and a few others that we will not name here - were identified by Gestaltists as deterministic factors in perceptual organization[68]. For example, consider the images in

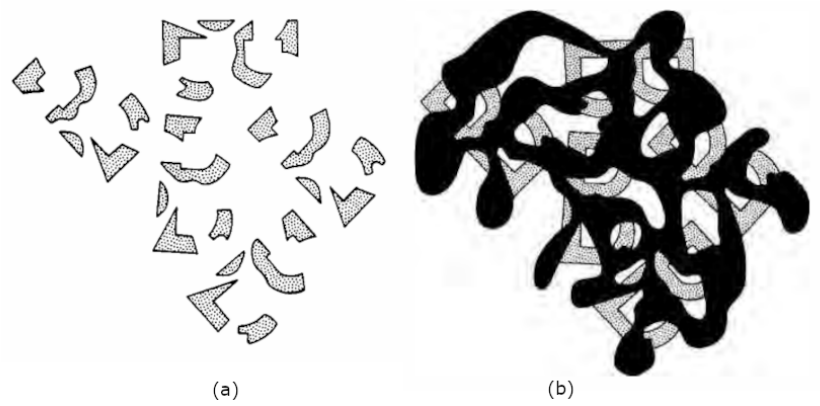

Figure 1.4 An example of the Gestalt property of closure; in (a), the individual pieces seem unconnected, while in (b), they seem to be capital letter "B"s. From [14].

Figure 1.4; these images were used by Bregman to illustrate the Gestalt property of closure. In image "a", there is not enough information for human brains to try to close and connect the (seemingly) random shapes. In image "b" though, we can't help but see the pieces as making up the five capital "B" shapes. Similarly, Bregman notes that when an audio stimulus is masked with a louder stimulus, the softer-sounding stimulus is still perceived to continue playing (even if it is explicitly removed).

1.1.2.3 The Use of Complex Sounds

Schutz and Gillard found that an overwhelming amount of research in non-speech audio perception relied on flat, simplistic tones, specifically, tones with a flat amplitude envelope [56]. They conclude that researchers choose these tones so that additional variables and complexity aren't introduced into their works, and to the reproducibility of their work. However, Schutz argues that these synthesized tones do not elicit the same perceptual responses as more complex tones. He notes that amplitude envelope and timbre allow humans to deduce information such as the materials used to create a sound, and even to detect events (he uses the example of a bottle breaking). The authors' early works relied on pure tones, usually simple sinusoids with a flat onset to offset response. That has since changed to synthesized instruments with a more complex amplitude envelope and additional timbre features.

Furthermore, studies have shown that the psychoacoustic properties of sounds need to

adhere to listeners' predefined mental schemas. For instance, Walker found that in magnitude estimation tasks (i.e. changes in stock prices, velocities, pressure, etc.), existing schemas on the polarity of values were an important design consideration [69]. In particular, he found that listeners sometimes perceived an increase in a modulated psychoacoustic property to an increase in magnitude, but other times related a decrease in magnitude to an increase in some property. He warned that sonification designers need to evaluate such mappings before designing a sonification. Ferguson and Brewster performed studies showing that modulation of psychoacoustic properties that generally have a negative connotation, such as noise or roughness, may be better at conveying properties in sonifications that are considered negative, such as stress or pressure [31].

1.1.3 Psychoacoustic Properties

The properties of sound and how they are interpreted by listeners are called **psychoacoustics**. There are many properties of sound that may be manipulated for the transmission of information. Dubus and Bresin reviewed a large body of sonification research and identified 30 properties and sub-properties that were being used (Table 1.3)[28].

"[P]erceived relations" in the aforementioned definition of sonification implies that the application of a particular sonification may be interpreted differently by different users. Walker and Kramer found that when mappings from some data dimension to a sound property match the listeners' expectations, that performance increased [70, 32]. For example, multiple studies have discovered that listeners may perceive a rising pitch as indicative of a positive or a negative change in a corresponding data element. In user studies on the sonification of magnitude estimation of temperature, pressure, size, and weight, researchers found that listeners preferred a negative scale when presenting weight increases by pitch [70]. In another study, researchers created mappings to convey blurriness of astronomical images and found that some users perceived an increase in resolution as a positive mapping and others as negative [30]. It seems then that allowing users to invert polarity could render a sonification more useful to a wider audience.

Pitch
Pitch Range
Timbre
Instrumentation
Polyphonic Content
Voice Gender
Allophone
Spectral Power
Amplitude of Harmonic
Frequency of Harmonic
Roughness
Brightness
Center frequency of filter
Saliency
Dynamic Loudness
Spatialization
Doppler Effect
Tempo
Duration (Rhythmic duration, Event duration, Ambient duration, Non-specified duration scale)
Sequential Position
Melody Lead
Articulation
Decay Time
Melody
Harmony
Chord Progression
Spectral Duration
Reverberation Time
Performance Activity Level

Table 1.3 Psychoacoustic Properties used for information presentation in prior literature, as identified by Dubus and Bresin [28]. The authors of that paper discovered that the use of pitch was used significantly more often that the other properties.

1.2 Related Work

There is not a great deal of research on conveying class diagrams and related graph types in alternative forms. What follows are descriptions of the works that have influenced this work the most.

1.2.1 Student Struggles with UML

As we have already noted, students with visual impairments or blindness cannot use UML or are greatly hindered in their ability to use it. However, the use of UML provides a much better understanding of the structure of systems than simply reading the code. However, even students who report no visual difficulties struggle with UML concepts.

However, learning to code with readily available software structure visualizations has been shown to lead to better learning outcomes, as has the use of a model-driven paradigm to create software [77, 76]. More formal methods, such as Model-Driven Software Development (MDSD) encourage modeling throughout the development phase [23]. Although little pedagogical research has yet been done on MDSD and student learning outcomes, it is an area that holds promise.

A recent study examined the types of errors that students make when creating UML diagrams [22]. During the course of a semester, the researchers analyzed more than 2,700 UML diagrams submitted by over 120 students. They classified the types of errors the students made. In class diagrams, researchers found that students most commonly left out class operations and their arguments, forgot to include dependencies, and left out or incorrectly typed functions. A tool with multi-modal presentation capabilities of the relationships between objects may be able to help students better understand how to model (and therefore develop) software.

Reuter et al. studied student learning through a "think-aloud" method, having students describe their mental processes while working on UML modeling problems [55]. Their results indicate that UML complexity combined with the lack of appropriate learning scaffolds such as additional examples of UML solutions, as well as the need for additional tools to create

diagrams, are the primary causes of student struggles.

1.2.2 Adapting UML for Individuals with Visual Impairments

Existing work on adapting diagrams for people with visual disabilities and blindness is primarily based on text, tactile, audio, or mixed methods.

1.2.2.1 Tactile Methods

Several authors have detailed their efforts to create touch-based UML solutions. These efforts range from laborious, human-centered methods to expensive digital mechanisms. Although digitally created tactile methods are likely to be useful once the assistive technologies that could support them become cheaper and more prevalent, they currently suffer from the need for expensive, hard-to-find hardware.

A 2006 paper detailed the method of one professor in conveying class diagrams with note cards, cut plastic strips, and pushpins for a student with visual impairments [17]. The professor adapted the diagramming tasks by creating the same diagrams with the aforementioned supplies. Although a rigorous study was not performed, anecdotally, the professor noted that the errors the student made in an individual diagramming task were the same as those of peer students. In addition, the professor noted that the student found the system more useful than the audio-based solutions he had used before. However, the work did not detail how different types of relationships might be conveyed using this mechanism.

The 3D System Touch (formerly called Omni Phantom) [7] devices purport to recreate digital objects so that users can touch them as if they were real (Figure 1.5. Eid et al. studied the use of a software program to create UML diagrams called UML CASE, both with a mouse and keyboard and with an Omni Phantom device [29]. The system gave users the perception of holding a class; the more members a class contained, the heavier the class would feel. Diagrams can be edited by grabbing classes and dragging them around the diagram. Force responses would indicate when lines crossed or classes collided with one another, and an elastic effect was portrayed when a relationship between classes existed.

[7] https://www.3dsystems.com/haptics

Figure 1.5 3D Systems Touch devices create the impression that users can touch a digital object. Such a device was used in the UML CASE tool [29].

The researchers found that users took more time to build a diagram with the haptic device compared to the mouse and keyboard setup (5.2 minutes average time compared to 4.1 minutes). However, users overwhelmingly reported that the haptic mechanism improved the interaction. Unfortunately, the haptic devices used in this study are expensive and rare. A mechanism that uses a more ubiquitous technology is needed.

Refreshable Braille displays employ a grid of pins that may be raised or lowered to recreate Braille lettering and other shapes and symbols. Loitsch and Weber used this type of display to recreate UML diagrams created with the Microsoft Visio tool [41]. Unlike the previously cited works, Loitsch and Weber conducted a formal user study of their work. They recruited 7 computer scientists with visual impairments and asked them to use the tool for two different tests. In the first test, users were asked to examine the diagrams and then asked questions about the number and types of objects in the diagrams. In the second test, users were asked to examine the diagrams and answer questions about diagram structures. A NASA Task Load Index (TLX) [8] was performed to measure the cognitive load of using

[8]https://human-factors.arc.nasa.gov/groups/TLX/index.php 18

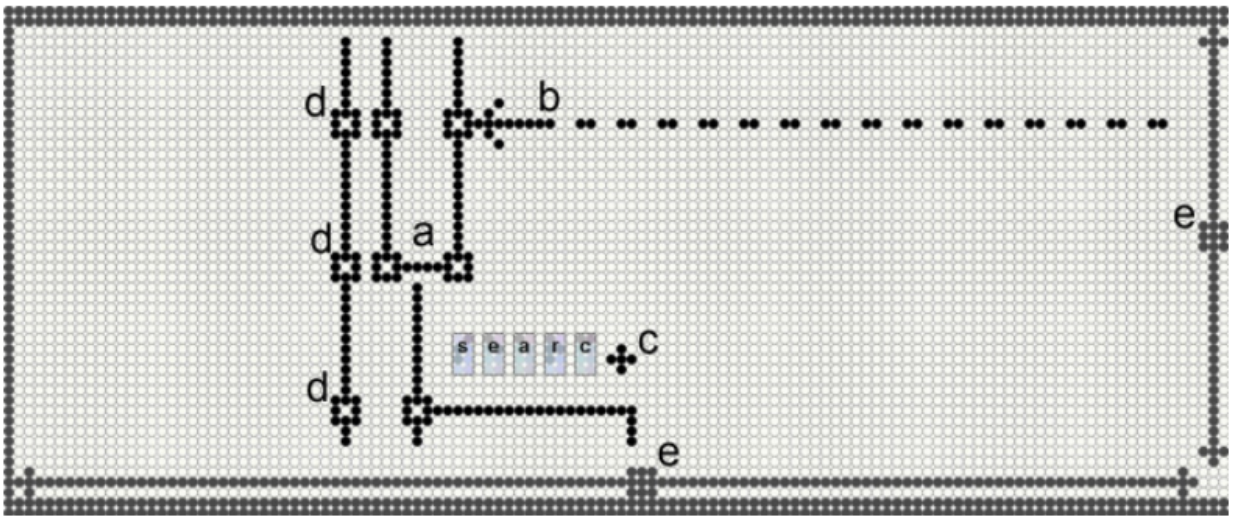

Figure 1.6 An example of tactile UML. The display media is a refreshable Braille display comprised of 120 columns and 60 rows of pins. Image reproduced from [41].

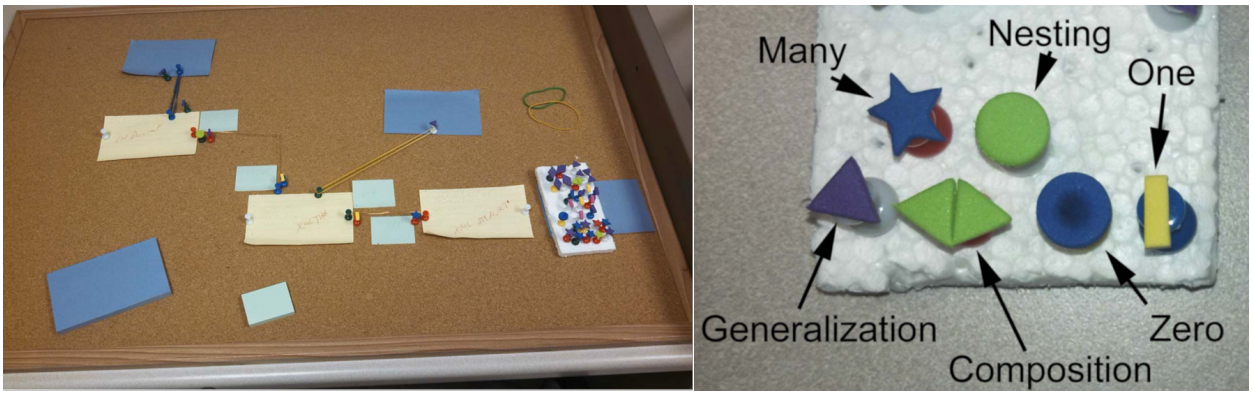

Figure 1.7 Tactile UML of Class Diagrams by Owen, Coburn, and Castor [48]. The image on the left is the corkboard "scene", populated with embossed Post-it notes, while the image on the right is an example of the foam pieces used to represent relationship type and multiplicity.

the tool. Although results were generally good, the tool suffers from the need for expensive, non-ubiquitous Braille displays.

Owen, Coburn and Castor used similar mechanisms in their 2014 work [48]. In this experience report, the authors detail a method of representing class diagrams via post-it notes representing classes, pinned to a corkboard. The notes were embossed with Braille to provide information about the classes and the class members. Wire or rubber bands were used to connect the classes. Foam shapes readily available in craft stores were used to

represent different types of relationships between classes and multiplicity (Figure 1.7). For example, a foam triangle indicates that the relationship is a generalization. Although this work shows promise, in that it is both flexible and provides complete artifacts (i.e., unlike the Brookshire method, it can convey relationship types and member information), it requires too much time to create a diagram. In the classroom, it is not unreasonable to expect that multiple diagrams may be needed for a single lecture, which would add considerable preparation time for the lecturer.

Doherty and Cheng developed a proof-of-concept mechanism for 3D printing tactile representations of UML diagrams, called PRISCA [27]. This system parsed diagrams from the *Visual Paradigm* software package. It used open-source software to create 3D CAD models that were then printed on a (relatively) inexpensive 3D printer (Figure 1.8). The major benefit of their mechanism is that diagrams do not need to be adapted manually for end users as a diagram would be printed exactly as drawn. This aids cooperation between coders who do not have visual impairments and those who do, as the same diagrams may be used for both populations.

Furthermore, the system can produce many types of UML diagrams. However, their mechanism has multiple downsides. First, even though the cost of 3D printers is decreasing, they are still somewhat expensive (Doherty and Chen mentioned that their model has cost $3,000), and they can be complex to use. Furthermore, the time to print a diagram was approximately 1.5 - 2 hours, making this method unsuitable for any rapid or agile design. Although this mechanism may be useful for conveying already designed systems, it excludes the inclusion of persons with visual disabilities in the design process.

1.2.2.2 Textual Methods

Some works have focused on creating text-based UML diagrams. The reasons for text-based UML are varied and often aren't for increasing access to diagrams for persons with disabilities, but rather to facilitate communication on messageboards, wikis, and through email [71, 51]. However, a text-based diagram may be more accessible as it is more likely to

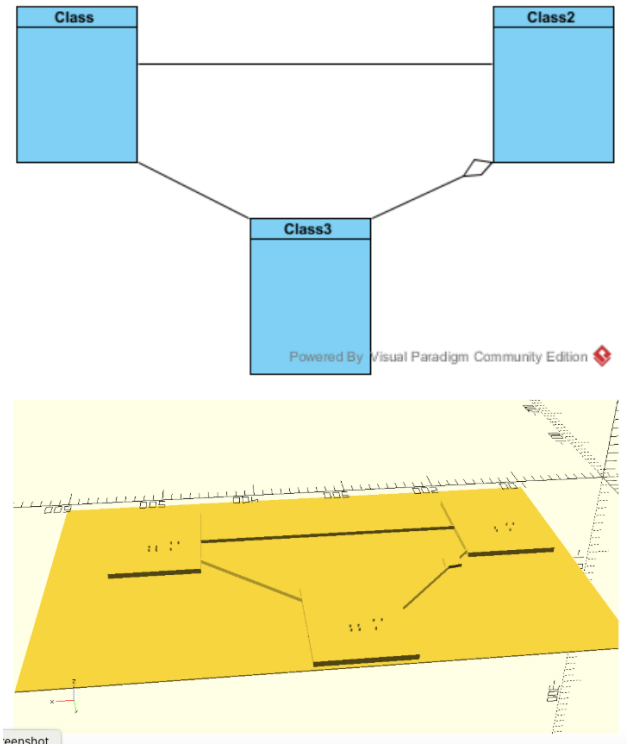

Figure 1.8 Illustration of the PRISCA system from [27]. The diagram is recreated with graph elements raised, providing a tactile interface for discovering elements and relationships between them. Figure reproduced from the original work.

include usable information for screen readers.

Washizaki et al. developed a text-based UML class diagram notation to convey these diagrams in domains that exclude graphical communications [71]. In their paper, the authors note that their system is beneficial for sharing models on mailing lists and over email. The character set was a subset of the ASCII standard[9]. Lines were drawn using hyphen, pipe, and colon, the selection of which conveyed the type of relationship. Unlike traditional class diagrams, the TCD method separates the class definitions (which include a list of the variables and methods of a class) from the relationship view (see Figure 1.9). This design choice limits the amount of information that a reader must consume at once, which could make this mechanism more useful for use with assistive technologies such as screen readers. However, this solution does not scale well and still uses visual mechanics such as lines to represent relationships. A large diagram, or one in which relationship lines cross, would be

[9]https://webstore.ansi.org/standards/incits/ansiincits1986r2002

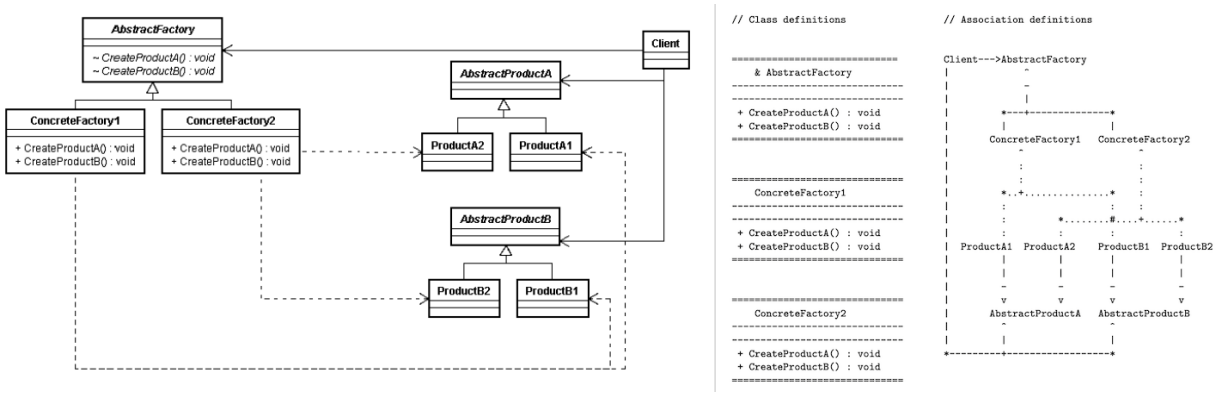

Figure 1.9 A class diagram (left) represented using the Text-based Class Diagram (TCD) method (right). An important feature of TCD is the separation of a class definition from the relationship hierarchy between classes. Figure from the original work [71].

no more consumable to assistive technologies than a graphical diagram.

PlantUML[10] is a robust, succinct, text-based mechanism for conveying a large variety of UML diagram types (we will focus here only on class diagrams). The designers of this system created it to be embeddable within the source code, and a Java-based tool is provided to parse the UML definitions from the Java source code. The diagram specifications must be enclosed with the tags `@startuml` and `@enduml`. Classes are represented by simply typing the `class` keyword followed by the class name. Symbols may be added to a class in one of two ways; either by entering the class name followed by a colon and the symbol name or by enclosing the symbol names within curly brackets, following the class declaration 1.10. Methods are specified by the symbol name followed by a set of parentheses. A variety of arrow types can be produced through combinations of line types and arrowheads, all conveyed with text. For instance, `<|--` between `Student` and `ComputingStudent` represents a solid line with a closed but unfilled arrowhead pointing from the second class to the first. The syntax of the mechanism provides facilities to represent access modifiers (public, private, protected, and default), stereotypes, notes, and other UML elements. Graphical diagrams can be automatically compiled from textual representations using the Java-based tool provided by the PlantUML authors. PlantUML has been used successfully in an educational setting by a

[10]https://plantuml.com/

```
@startuml
 class Student
 Student : int idNumber
 Student : String name
 Student : float gpa
 Student : boolean equals(Student)
 Student : String toString()
@enduml
```

```
@startuml
 class Student {
      int idNumber
      String name
      float gpa
      boolean equals(Student)
      String toString()
 }
@enduml
```

Figure 1.10 Two examples of a PlantUML class specification; the first uses the colon syntax, while the second uses curly braces to denote ownership of symbols.

student with vision impairment to create diagrams[19]. However, the tool does not provide any facilities for exploring existing diagrams.

Earl Grey is a similar text-based UML language, similar to PlantUML [43]. Earl Grey differs from PlantUML in that it is designed to break completely from graphical methods of element and relationship conveyance. Although PlantUML is text-based, it relies on the use of symbols constructed from the ASCII character set to mirror visual UML elements such as solid and dashed lines and a variety of arrow types. The Earl Grey designers believe that such mirroring could lead to confusion for diagram consumers, and instead opted to use words to express these concepts. For example, they rely on the specific use of the terms "aggregation" and "association" to denote association types and use the term "isA" to convey generalizations. The classes are defined with their name and symbols within the `class` and `end` keywords (Figure 1.11). Although Earl Grey is designed for educational use, even a diagram with relatively few relationships could result in a great deal of text that must be consumed. As no tool for searching or exploring a diagram written in this language is provided, it is unlikely to be very useful in all but the simplest of scenarios.

1.2.2.3 Audio Methods

Metatla et al. noted that UML diagrams are hierarchical structures that present the same information in several different ways [44]. For example, they noted that a diagram consisting of a class A and B, with an association between them, that a spoken representation of that information might be from the perspective of A or B (i.e. "A holds an instance of B", or

```
class Color
  red : int
  green : int
end
class Shape
  /*...*/
end
class Point isA Shape
end
class ComposedShape isA Shape
  /*...*/
end
```

```
aggregation
  0..1 ComposedShape /*start*/
  0..* Shape /*end*/
end

association
  0..* Shape
  1 Color
end
```

Figure 1.11 The Earl Grey text-based class diagram notation. Generalizations are indicated by the "isA" keyword, while other relationship types are denoted with the relationship type, starting and ending classes, and the "end" keyword. Multiplicity may be added to the starting and ending class entries.

could be from the perspective of the relationship (i.e. "an association exists between the two classes". They designed an auditory display for class diagrams that emphasized the various perspectives. Their design had menus for objects, associations, and generalizations (they did not focus on other relationship types). The objects menu contained a list of all classes in the diagram, while associations and generalizations contained lists of those relationship types and the classes that comprised them. The designers presented information in two ways: a "verbose" mode that used text-to-speech descriptions, and a "terse" mode that used some text-to-speech combined with non-speech audio. Non-speech audio is used to represent a relationship, with different timbres used to represent different types of relationship. The stimulus consisted of one short tone and one long tone, or one long tone followed by one short tone. In comparing the tones with the visual element of a line ending with an arrowhead in classical UML, the short tone was representative of the arrowhead, while the long tone represented the line portion. If we had two classes, A and B, and the direction of the relationship was from B to A, the designers would reflect that with the class name "A", a short tone (an arrowhead pointing at A), a long tone (representing the connecting line) and the class name "B". If the direction were A to B, the short and long tones would be presented in reverse order. In addition, they used amplitude modulation to create what they

called a "coming into" or "coming from" effect. An experiment was performed to measure the effectiveness of both verbose and terse strategies. Participants were asked to answer questions about several diagrams. Ten trials were conducted using the verbose mode, and an additional ten for the terse mode. The participants scored well, with a mean score of $\mu = 96$ ($\sigma = 6.08$) out of 100. A statistically significant time difference was not found between the completion times for the two presentation modes; this is an important result, showing that non-speech audio can effectively replace spoken descriptions of relationships in these diagrams. This is reinforced by the work we have since performed [75], as well as by the work presented later in this document.

Unfortunately, the work by Metatla did not include specific information on the audio used. They note the use of "Different timbres", as well as amplitude modulation, but do not provide any more information than that. Furthermore, the inability to reflect a holistic view of any part of the diagram at once is cumbersome. The appeal of visualizations such as UML is that the consumer can focus their attention on the macro view to notice patterns and trends, and then "drill-down" to specific parts of the diagram for further analysis. By including only three hierarchies, each of which provides merely a focused view on a particular type of element or relationship type, a consumer is forced to jump amongst the various menus to develop a complete schema of any single element. A better mechanism would incorporate this fine-grained view that Metatla's work provides with an additional macro-level view or views, as well as a mechanism to search for elements and quickly navigate between them.

1.2.3 Mixed-mode Methods

A method for presenting node link graphs, dubbed "TADA", was published in 2024. This software relies on touch and speech for input and produces speech and musical notes to produce information about node-link diagrams (of which UML class diagrams can be considered a subset). The software was made to work with commodity tablets, without additional special hardware. The sounds are played as the user moves their hand across the tablet surface, and the system provides seven different interaction techniques. A 150Hz

pure tone indicates to the user that their hand is near a diagram element (a node or a link). The software represents a node in the graph presented with a French horn sound. The number of connections the node has influences the pitch of the French horn - the more connections the node has, the higher the pitch. A plucked guitar sound represents links between nodes; the shorter the connection length, the higher the pitch. An experiment was carried out with 25 participants (14 female, 10 male, 1 transgender, all of whom self-reported as legally blind. Of this group, 20 were unable to visually perceive any part of the diagrams on the tablets, and five were able to visually perceive only varying amounts. Participants participated in six blocks of activities. In each block, a training phase was completed before a testing phase. Participants were asked to answer questions about the diagrams presented to them. Participants were able to accurately answer questions about the diagrams, though nearly a fourth of them needed assistance with more than one of the questions. The study designers completed a NASA-TLX task load index to determine the perceived workload of the participants using the system and found that for the majority of the participants, the perceived workload was low. However, for some of the participants, the physical and mental demands of producing some of the interaction techniques were high. Furthermore, the system suffers from the design choices of using a fixed-size graph and the inability to pan or zoom, making this only usable for the most simplistic graphs.

1.2.4 Identifying Student Struggles with UML

In order to better understand the user interface requirements for conveying class diagrams, we first consulted the literature to determine the issues that cause students to struggle with class diagrams. We felt that issues that cause a student to struggle could potentially be exacerbated by presentation via audio. Although the literature was helpful (and is summarized in Section 1.2), we discovered a knowledge deficit. Studies often examined the results of student work, but we did not find publications asking students to relay their perceptions of why they struggle.

Though it is often maligned, the Object Oriented (OO) programming paradigm remains in very high use in industry, and is taught in many undergraduate computing courses. Although exact numbers describing use of the paradigm do not exist, we do know that languages that support OO are consistently ranked highly in lists of the most used programming languages. At the time of writing, both the *IEEE Spectrum*'s 2023 Top Programming Language list and the PYPL Popularity of Programming Language list show the first six of their top languages support OO, and one language (Java, the second highest rated in both) requires the OO paradigm [20, 8]. The TIOBE index provides similar results, with seven of the top eight languages supporting OO [9]. However, undergraduate struggles with learning OO concepts are long-standing and well documented [76].

Although there are other ways of visually modeling software such as Entity-Relationship Diagrams (ERDs), Class-responsibility-collaboration cards (CRCs), and even informal drawings, class diagrams are the commonly accepted standard for modeling software across most undergraduate curricula. Unfortunately, undergraduate students continue to struggle with learning and using UML; a review of the literature by Muoz et al. identified four areas in which students struggle with learning UML [47]. In that work, they note that inexperience in cognitive processing, UML complexity, abstract conceptualization, and lack of feedback are the primary problems. In this work, we posit that students may additionally be experiencing a lack of "buy-in" due to inconsistent UML use across undergraduate courses, and a

misconception by students that since UML isn't used as much in industry, that it holds little value. Previous work has shown that UML is not as widely accepted in industry as it is in academia, with surveys by multiple industry professionals reporting that more than 70% of the respondents do not use UML in practice [38, 53]. In one work, frequently cited reasons for the lack of industry use included the complexity of the language and the perception of the need to commit to its use consistently to enjoy the benefits [52].

In this work, we survey students' experiences and perceptions of learning using UML class diagrams. The results do not indicate demographic differences in the perceptions of the students of their abilities, nor is there a significant correlation between the amount of time the students reported receiving UML instruction and their feelings of competency. Students ranked a need for more consistent use of UML throughout the undergraduate curriculum as the primary strategy needed to become more comfortable with class diagrams.

1.2.5 Study Design

The research team on this project has been based at Michigan State University (MSU) and Grand Valley State University (GVSU). MSU is classified as an R1, or Doctoral university with very high research output. GVSU is classified as M1, or a Master's Large university. Both universities' programs in Computer Science (the authors' field of study) are accredited through the Accreditation Board for Engineering and Technology, Inc. (ABET).

We created a set of survey questions common to both universities and then created specific versions for each university. This allowed the researchers to compare the two populations to see if specific trends existed across both groups, and to allow the researchers to see which specific classes at their universities were using UML Class Diagrams.

1.2.6 Research Questions

We identified the following research questions:

RQ1: Are there significant differences between the two universities that affect student comfort with UML?

RQ2: Are there demographic differences between students that affect student comfort

with UML?

RQ3: Do students report struggling more with using UML to design a software solution, or with implementing a coding solution from UML?

RQ4: Which factors do students indicate would be more helpful in becoming proficient with UML?

1.2.6.1 Population

The surveys were sent by email to undergraduates from both universities. There were no benefits or consequences for students to take the surveys. Between the two locations, 105 students responded. Of those 105, $n = 67$ completed all survey questions ($n = 38$ for MSU, $n = 29$ for GVSU). Of the 67, 89.6% ($n = 60$), reported an age in the range of 16-24, 9% ($n = 6$), in the range of 24-34, and 1.4% ($n = 1$) in the range of 35-44.

The standing of the class was also examined. The respondents were mostly upper class students (over 85.0%), with seniors making up 55.2% ($n = 37$) of the respondents, followed by juniors who made up 29.9% ($n = 20$). The freshmen respondents comprised 6.0% ($n = 4$) and the sophomores comprised 3.0% ($n = 2$). The remaining 6.0% ($n = 4$) responses indicated they were graduate students; this is due to the fact that GVSU offers an accelerated five-year program for a combined bachelor and master degree.

1.2.6.2 Data Collection

We used the Qualtrics Flexible Survey Tool (Qualtrics, Provo, UT) for data collection. Demographic data was collected under the human subject control protocols of both universities with the approval of both Institutional Review Boards.

Data were downloaded from Qualtrics as comma-separated value files and combined into a single file for analysis. Python was used to further clean the data, including selecting only completed surveys from the original 105, pre-processing and label encoding the data, and performance of all data processing.

Students were asked to rate their level of comfort with both designing a software solution with UML and implementing a software solution from an existing UML diagram. A

five-point Likert scale was used to collect responses, with possible values being Extremely Uncomfortable, Somewhat Uncomfortable, Neither Comfortable of Uncomfortable, Somewhat Comfortable, and Extremely Comfortable.

The students were then given a list of seven strategies that could improve their comfort level with UML and were asked to rank them in order of usefulness. Each strategy required a response, and no two responses could be assigned the same value. The students were also given an open option to suggest their own strategies.

1.2.7 Results

1.2.7.1 RQ1: Are there significant differences between the two universities that affect student comfort with UML?

The goal of this work is to determine whether there are changes to how class diagrams are taught that undergraduates feel would be beneficial to their understanding and comfort with designing and implementing software with them. We first needed to ensure that the differences between the way UML was taught in each university did not have a significant effect on the comfort levels reported by the students. We identified several areas in which GVSU and MSU may have taught UML differently and analyzed whether the differences led to statistically significant differences in the reported levels of comfort with these diagrams.

First, we asked students to report the approximate number of hours of instruction they received explicitly in learning UML Class Diagrams. The choices were <1, 2-5, 5-7, 7-10, or 10+ hours of instruction. The number of responses for each selection and each university was calculated and a Chi-square test was performed to determine the likelihood of differences between the two groups. The resulting value $p = 0.06$, while greater than $p = 0.5$ is too small to consider significant, particularly with a single sample from each organization. Furthermore, a Cramer's V test found only very small effect sizes between time of instruction and reported comfort levels ($\phi_c = 0.294$ for comfort designing solutions with class diagrams and $\phi_c = 0.298$ for comfort implementing solutions from them). In the best case, there seems to exist only a weak correlation between the reported amount of time students receive

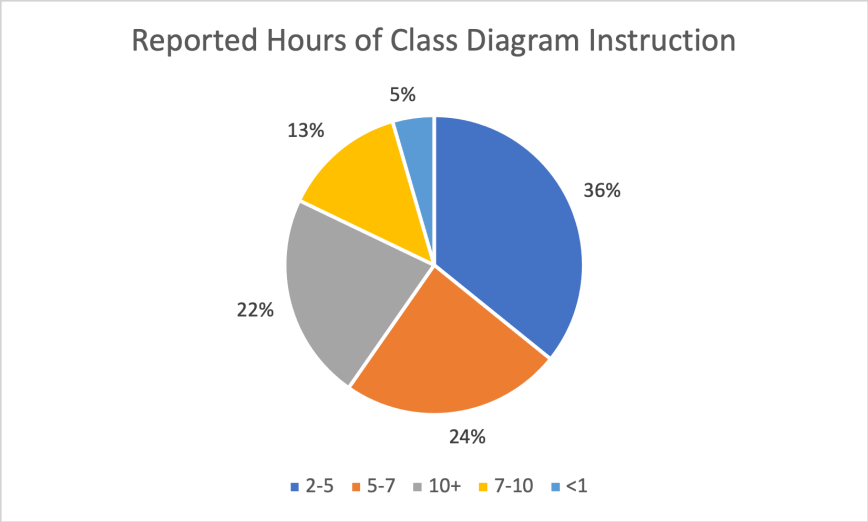

Figure 1.12 Students were asked to approximate the range of the number of hours of class diagram instruction they had received. This chart shows the percentage of students that responded with each range.

Course Topic	**Number of Responses (%)**
Software Engineering	53 (79.1%)
Introductory Programming (CS1 or CS2)	23 (34.3%)
Database	17 (25.4%)
Systems Analysis and Design	6 (9.0%)

Table 1.4 Course types identified by students as providing explicit UML class diagram instruction.

instruction and student understanding of class diagrams and the concepts they represent.

Students were asked to specify in which courses they received explicit UML class diagram instruction. The courses were then categorized and grouped by topic. Differences in teaching styles and instructor topic preferences mean that many different courses were reported. However, we don't find it meaningful to include outlier class types. As a comprehensive analysis broken down by instructor is far beyond the scope of this paper, we decided to accept a response if the same class type was indicated as being a source of class diagram instruction by more than 3 of the respondents from that given university. The results indicate that class diagram instruction occurs primarily in four course types: Introductory (CS1 and CS2), Systems Design and Analysis, Database and Software Engineering courses (Table 1.4).

The students were then asked to report the percentage of classes in their major in which

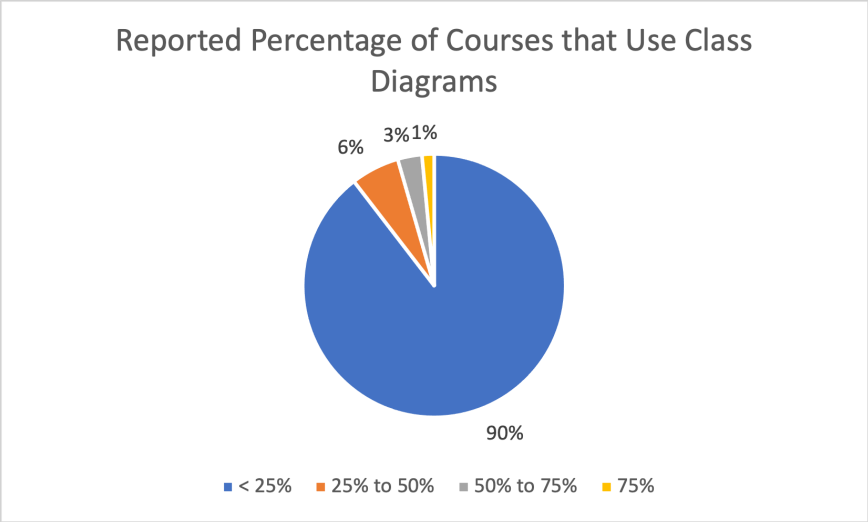

Figure 1.13 The percentage of courses students reported that use class diagrams. Over 90% of respondents state that less than a quareter of their computing courses use class diagrams.

they *used* class diagrams. This was not to measure instruction time in class diagrams, but rather to determine how many courses reinforced the importance of them by having students use them to design solutions. Although class diagrams are likely not useful for all courses (we have a hard time thinking of a use for them in a low-level course on architecture and hardware, for instance), there are many software engineering and computer science classes that require programming assignments that could be modeled with class diagrams. However, the overwhelming majority of the two groups of students reported that $< 25\%$ of their major classes used class diagrams - 82.8% ($n = 24$) for GVSU and 94.7% ($n = 36$) for MSU (Figure 1.13).

We then asked the students to identify the specific courses in which they used UML. Again, we collected and coded the courses by topic and calculated the aggregate results. Course topics that had less than 3 respondents were again dropped as outliers. The responses to this question were nearly identical to those identifying courses that explicitly taught UML, with one additional topic - Mobile Application Development.

Course Topic	Number of Responses	Percentage of Respondents
Software Engineering	55	79.1%
Introductory Programming (CS1 or CS2)	19	34.3%
Database	13	25.4%
Systems Analysis and Design	6	9.0%
Mobile Application Development	6	9.0%

Table 1.5 Course types identified by students as requiring the use of UML class diagrams.

	Gender	Age	Ethnicity	HaveDisability	PrimaryLanguage	ComfortableDesigning	ComfortableImplementing
Gender	1.000000	0.180230	0.254728	0.171008	0.150290	0.228968	0.127799
Age	0.180230	1.000000	0.227471	0.194302	0.279974	0.194936	0.253712
Ethnicity	0.254728	0.227471	1.000000	0.377822	0.352537	0.279527	0.278385
HaveDisability	0.171008	0.194302	0.377822	1.000000	0.081541	0.328494	0.255631
PrimaryLanguage	0.150290	0.279974	0.352537	0.081541	1.000000	0.162019	0.138819
ComfortableDesigning	0.228968	0.194936	0.279527	0.328494	0.162019	1.000000	0.519326
ComfortableImplementing	0.127799	0.253712	0.278385	0.255631	0.138819	0.519326	1.000000

Table 1.6 : Contingency table of Cramer's V effect sizes between demographic variables. No two demographic variables were highly associated with student learning outcomes.

1.2.7.2 RQ2: Are there demographic differences between students that affect student comfort with UML?

We created a contingency table using the Cramér V test in the demographic fields of gender, age, ethnicity, primary language, and comfort levels reported for both the design and implementation of UML solutions (Table 1.6).

For gender, 73.1% ($n = 49$) reported identifying as male, 20.9% ($n = 14$) as female, 4.5% ($n = 3$) preferred not to answer, and 1.5% ($n = 1$) reported identifying as nonbinary. Responses to questions about ethnicity found 73.1% ($n = 49$) identified as White/Caucasian, 10.4% ($n = 7$) as Asian - Eastern, 6% ($n = 4$) as Asian - Indian, 3% ($n = 2$) as African-American, 3% ($n = 2$) as Hispanic, 3% ($n = 2$) as Mixed race and 1.5% ($n = 1$) as Middle Eastern.

Both universities attract a large number of international students. To ensure that lan-

guage barriers were not the primary cause of student UML struggles, we asked students to select their primary language. Most of the students ($95.5\%, n = 64$) listed English as their primary language. The other languages listed were Arabic, Mandarin and Marathi ($1.5\%, n = 1$) each.

Students were also asked to self-report whether they considered themselves to have a disability and if so what type of disability, as researchers wanted to ensure that visual or hearing impairments were not a major factor in student struggles. Of the 67 students, 70% ($n = 47$) reported no disability, 20.9% ($n = 14$) reported some type of disability, 7.5% responded with "Maybe", and 1.5% ($n = 1$) preferred not to say.

There were no demographic variables strongly associated with higher reported levels of comfort in using UML for software design or implementation.

1.2.7.3 RQ3: Do students report struggling more with the use of UML to design a software solution or with implementing a solution using UML?

When asked to choose their level of comfort in designing an acceptable solution to a software problem using UML, 37.3% ($n = 25$) students noted that they were "Somewhat comfortable". An additional 16.4% ($n = 11$) replied that they were "Extremely comfortable". The remaining students reported "Neither comfortable nor uncomfortable" at a rate of 16.4% ($n = 11$), "Somewhat uncomfortable" at 16.4% ($n = 11$), and "Extremely uncomfortable" at 7.5% ($n = 5$) (Figure 1.14).

Similarly, when ranking their comfort with implementing a software solution from a UML diagram, 44.8% ($n = 30$) the students were "Somewhat comfortable", and 10.4% ($n = 7$) were "Extremely comfortable". The remainder of the students reported more negative results, with 25.4% ($n = 17$) responding "Somewhat uncomfortable", 10.4% ($n = 7$) "Extremely uncomfortable", and 9.0% indicating that they were "Neither comfortable nor uncomfortable" with this topic (Figure 1.15).

A Cramer's V test reported a moderate association between comfort design and comfort implementation of a solution. The average numbers of responses were similar for the

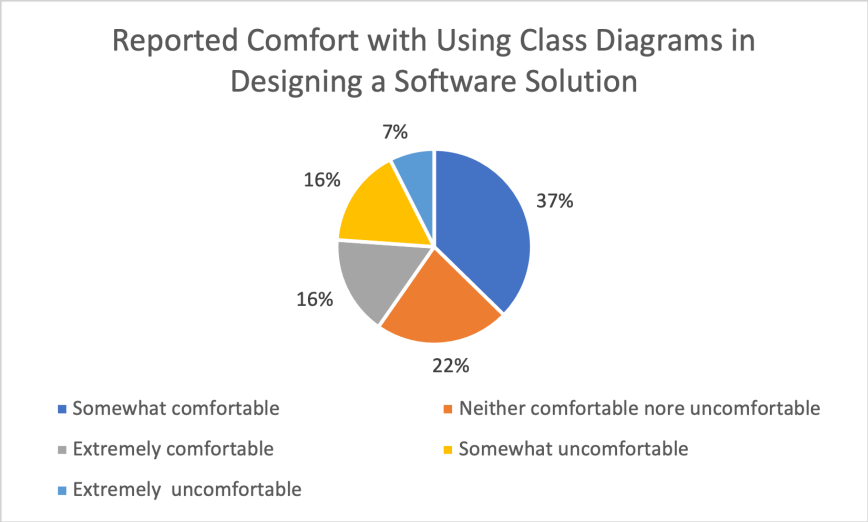

Figure 1.14 The reported levels of comfort students reported with using class diagrams to design a software solution.

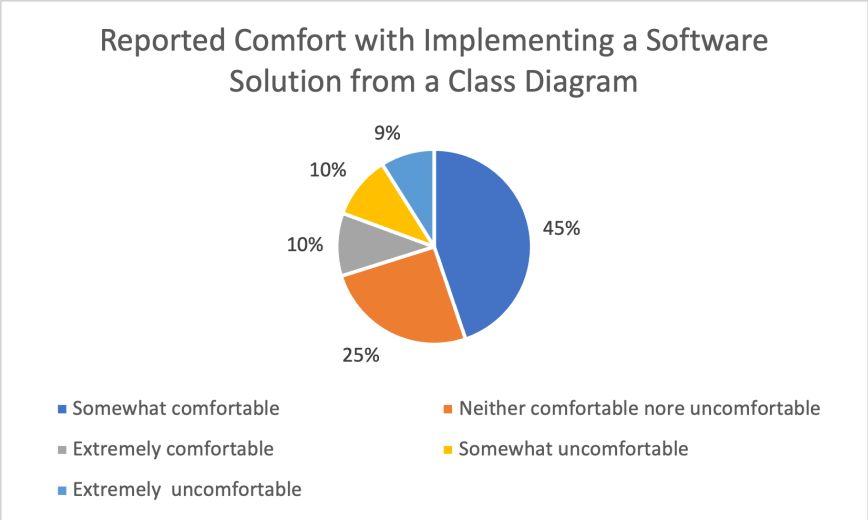

Figure 1.15 The reported levels of comfort students reported with using a given class diagram to implement a software solution.

"Somewhat comfortable" and "Extremely comfortable" responses, with 53.7% responding with some level of comfort in designing a solution and 55.2% feeling at least some level of comfort in implementing a solution. Although it is not surprising that those two categories are correlated, these findings indicate that nearly half of the students do not feel comfortable using class diagrams and that many feel uncomfortable doing the tasks.

Furthermore, 74.6% ($n = 50$) of the students responded that they do not use UML when it is not assigned.

1.2.7.4 RQ4: What factors do students indicate would be more helpful in becoming proficient with UML?

We asked students to rank the following strategies to improve student comfort with UML class diagrams:

1. More time spent on the process
2. Better UML resources for reference
3. More/Better design examples
4. More/Better implementation examples
5. Multiple modalities for exploring diagrams such as haptic or audio diagrams
6. Better creation/viewing software for diagrams
7. Better consistency in using UML in courses

Respondents were asked to rank each of these fields according to their perceived usefulness in increasing comfort with class diagrams. The students labeled each strategy with a value between 1 (most helpful) and 7 (least helpful). A weighted average was calculated for the number of responses for each strategy with the value assigned by the students 1.7.

The highest ranked strategy was "Better consistency in using class diagrams across the curriculum". The students also indicated a preference for "More time spent learning class diagrams", "Better design examples", and "Better implementation examples". When given the option to provide an open response, only three students did so. Their suggestions were "Lots more practice such as a program that grades UML you create after giving you example

Strategy	Weighted Average Rank
Better consistency in using class diagrams throughout the curriculum	11.250
More time spent learning class diagrams	10.179
Better design examples	10.071
Better implementation examples	9.857
Better class diagram references	9.426
Better diagramming software	9.392
Multiple modalities for exploring diagrams	8.250

Table 1.7 Weighted average values of the ranking of strategies to improve comfort with class diagrams.

questions", "[Course] is a lot of work and because of that I didn't do the best. With that said, I still feel as though more UML forced practice would be great," and "Using in more courses. I use UML for personal projects but never in class". These statements reinforce the results that students feel that more consistency and interaction with class diagrams would be helpful.

1.2.8 Discussion

The responses to the surveys of both universities indicate that students are receiving similar levels of instruction in both designing and implementing solutions. Both show similar numbers of hours of specific instruction given and similarity in the percentage of classes in the major that use class diagrams. However, this percentage is small.

We were surprised to find that the same courses that students identified as teaching UML class diagrams were the only ones that used them. Perhaps this is to be expected, but it almost certainly reinforces to our students that modeling software through the use of these diagrams lacks value. Most computer science and software engineering courses probably require students to write software, but according to these results, more than 75% of the courses do not ask students to model their thinking and design process through the use of the tool we teach them that exists for that purpose.

Furthermore, both GVSU and MSU require a collaborative culminating experience course in which students are expected to develop a comprehensive solution to a software problem,

yet *not a single student in our sample identified that course using class diagrams.* While it may be true that professional software developers don't need to model software and can instead rely on more agile development processes, we must keep in mind that students are still developing their skills and mental models, and thus need this extra reinforcement. In addition, student responses identify the desire for more consistent use of class diagrams throughout the curriculum as the most helpful strategy to improve their level of comfort with them.

As Large Language Models (LLMs) such as ChatGPT [11] and Github Autopilot [12] become increasingly capable of writing efficient code, it becomes ever more important that next-generation engineers are able to deconstruct and understand larger problems. Modeling throughout the undergraduate curriculum - and throughout an engineer's career - holds the potential for helping students architect solutions to increasingly complicated problems. The ability to use those models, to update them and modify them, and to then generate code from the formal specifications has the potential to create high-quality software with less human work.

[11] https://chat.openai.com/
[12] https://github.com/features/copilot

CHAPTER 2

DETERMINING USER PERCEPTION OF AUDIO STIMULI AS GEOMETRIC SHAPES

We began this work with the desire to convey different shape types with audio. We weren't specifically focused on class diagrams at this point; we had hoped to create a mechanism that would be useable for many different types of node-edge graphs, flow-charts, etc., reasoning that most visual graphs rely upon the mechanism of shape to present different types of entities. It was theorized that we could represent different shapes with spatial audio, by sequentially playing the spatialized points that made up the shapes.

2.0.1 Methodology

We began our work with the determination of the efficacy of conveying simple (six-sided or less), closed, convex shapes with sound. This work is summarized here, published in [73] and expanded in [74]. We chose closed, convex shapes, as many graphical diagrams use such simple mechanisms. We developed software that maps audio to an 18.1 cm^2 area. The math for the mapping made use of stereo panning and pitch manipulation using these functions:

$$Amplitude_{left}(x) = 1.0 - \frac{x}{660} \tag{2.1}$$

$$Amplitude_{right}(x) = \frac{x}{660} \tag{2.2}$$

$$Pitch(y) = y + 220 \tag{2.3}$$

Here x is a normalized volume value and y is in Hz. According to existing work, human hearing can discriminate between two frequencies below 4 kHz with precision down to 2 Hz [57]. However, above 4-5 kHz, our hearing is much less precise [13]. We selected our range based on these facts, combined with the desire to keep our participants as comfortable as possible during the testing; we felt that higher-frequency sounds could become irritating over long testing sessions.

We developed a Java application for presenting shapes through sound and for collecting data. The participants were shown 10 different shapes, each made up of six or fewer line

segments representing convex polygons. A convex polygon is a shape in which any line drawn between two points within the shape remains entirely inside the polygon. A Planar PCT2485 touch screen display was used to present the two-dimensional shapes, with a display area of 18.1 cm by 18.1 cm. Participants could use a Korg nanoPAD2 MIDI device to control the playback of the shapes' sonifications, including play/pause, speed, and the direction of playback (either clockwise or counterclockwise). The audio output was generated by a computer and routed through a Behringer HA4700 multichannel headphone amplifier, allowing each participant to individually adjust the volume. The researchers monitored the audio to ensure proper system functionality and the participants listened using Sony Professional MDR-7506 stereo headphones.

Participants were allowed as much time as they needed to practice and familiarize themselves with the setup. On-screen buttons were provided to play the audio for the center coordinates, values along the edges of the plane, and random points. During practice, participants could play a random point, select the perceived location on the touchscreen, and then compare it with the actual position. They could also play sounds for the sample shapes and view the corresponding visual output. When participants felt ready, they informed the researcher, who would then begin the presentation of the study shapes.

Each participant was first presented with five test shapes, all composed of line segments. A small click was played at the end of each line segment to indicate a change in direction. Participants could replay the shape as many times as they wanted, choose to play it clockwise or counterclockwise, adjust the speed of playback, and control the volume. Once confident that they understood the shape, participants would attempt to draw it on the basis of their perception, after which they were shown the correct shape. After five trials, the researcher informed them that the correct shapes would no longer be shown for the remaining 10 tests. After these final attempts, the session ended without revealing the shapes to prevent bias or prior knowledge in future participants.

The shapes included equilateral, isosceles, and scalene triangles, squares, rectangles, di-

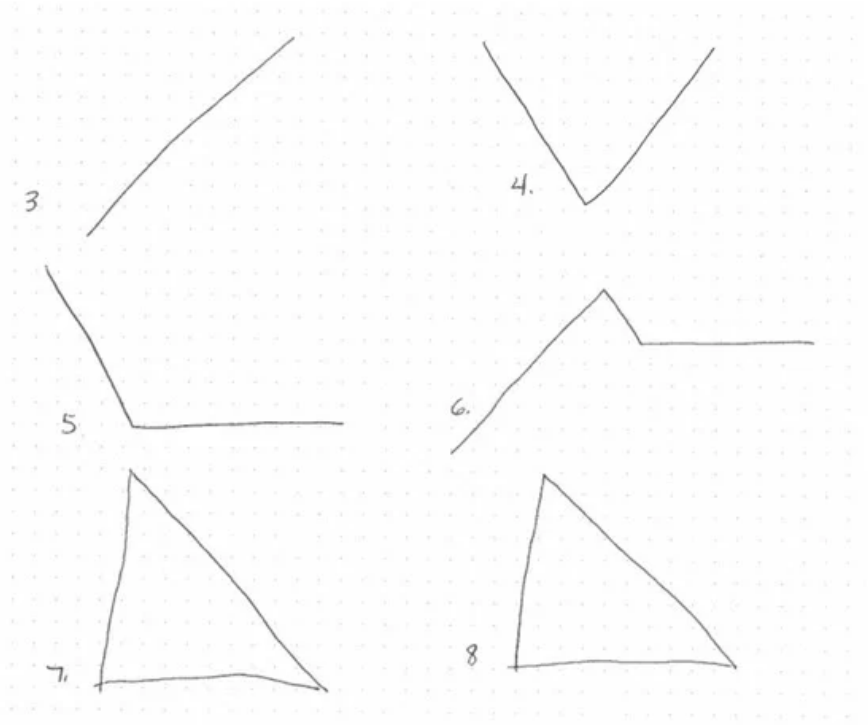

Figure 2.1 Sample result from our study on conveying simple shapes with audio. Taken from [74].

amonds, and randomly generated polygons composed of four to six line segments.

2.0.2 Results of the Shape Drawing Study

We recruited 66 students (16 female, 50 male), all undergraduates with an average age of $\mu = 21.05$ years ($\sigma = 2.8$ years), all of whom reported normal hearing and carried out our experiment. The results were poor. Five participants were unable to finish the task as they found it too complex. Although we explicitly told each participant that all shapes were closed and convex, showed them samples, and allowed them to practice before testing them, most of the responses did not meet the criteria. Instead, participants frequently created drawings with unconnected line segments (Figure 2.1).

Examining the pictures participants drew and conversing with the participants after testing led us to conclude that though the individual mappings were simple and easy to understand, the combination of the two led to imprecision. However, the researchers were able to use the system with decent accuracy, so we were not yet ready to abandon the idea.

We wanted to be sure that this system would not work and identified a possible lack of motivation as an influencing factor, as the task itself was not very interesting or exciting. We decided to repeat the study, but in a context that would be more fun and motivating for the participants. To do this, we developed a video game.

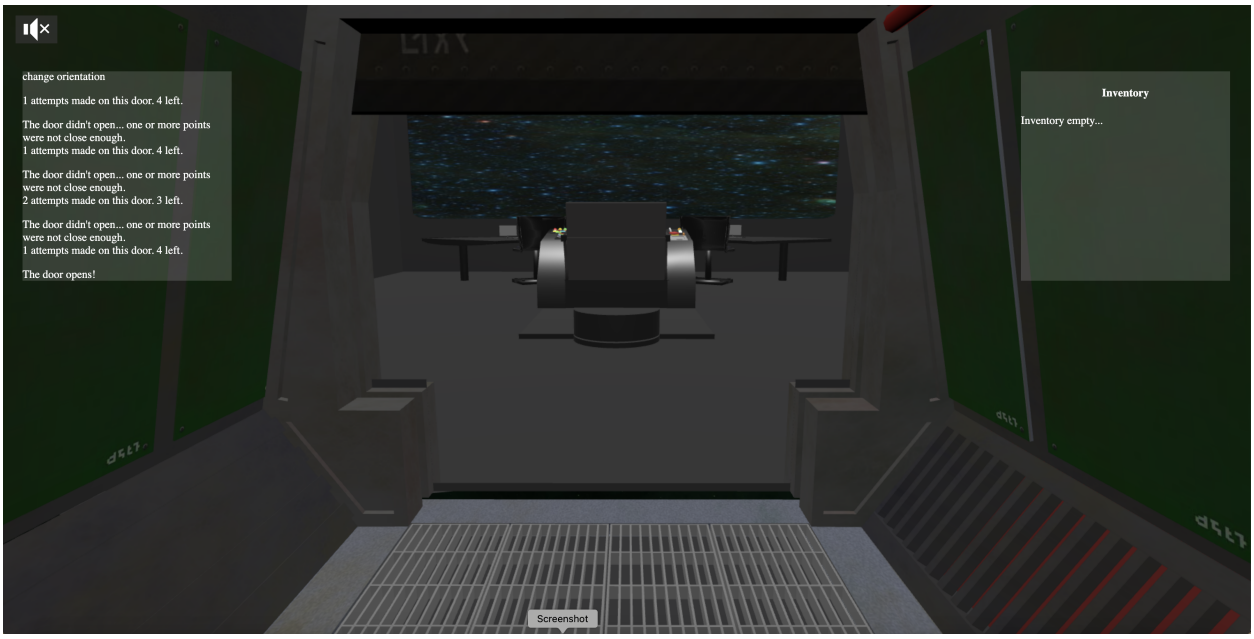

Figure 2.2 Sample scene from video game created for our study. In this image, the user has completed a spatial audio positioning task that unlocked the door.

2.0.3 Trying Again, with Increased Motivation

We were not ready to give up on the use of spatial audio to convey points in space. We felt that we were able to perform the tasks relatively well ourselves and that perhaps lack of motivation or interest with the task was the problem. We hypothesized that a similar task in the context of a video game might yield better results.

2.0.4 Methodology

We decided on a game where the player docks with and is trapped on an alien spacecraft. The premise of the game was that the aliens who created the ship did not use vision; instead, all controls are audible. The player's task was to explore the ship to find pieces that fixed a broken hanger door that they needed to open to escape. Most of the doors of the ship were locked, but could be opened by touching them in the locations communicated using spatial audio signals (Figure 2.3).

We acquired and created 3D assets for the spaceship, and built a prototype using the

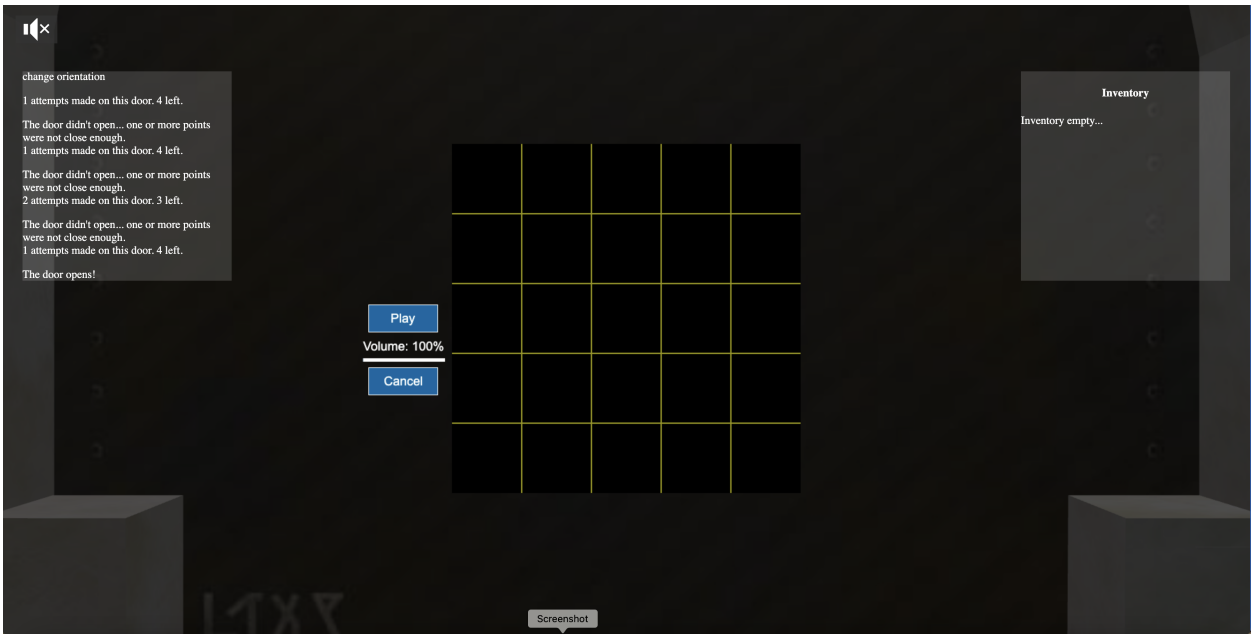

Figure 2.3 When players encountered a door, the game scene switched to a mini-game that conveyed points in a plane via spatialized sounds. Players clicked where they perceived the sounds to be on the plane. If the were precise enough, the door would unlock.

Babylon.js [1] Web game engine. The Howler.js [2] spatial audio library was used to spatialize our sounds. The stimulus used to convey the positions was a free 440Hz sine waveform found on Freesound.org [3]. The waveform was modulated by pitch using the equation 2.3 and panned by Howlerjs' equal-power panning function. We decided on this panning function as opposed to our original functions (Equations 2.1 and 2.2) in the hopes that it might lead to better results.

The researchers were able to play the game with reasonable accuracy, so we began preliminary testing with students. Unfortunately, even with the addition of the game context, the results were poor. We began preliminary trials, requiring users to be accurate within a 100 pixel distance from all conveyed points in order to unlock a door. This was not nearly enough, so we altered the task to use a 250 pixel difference. Again, this was not a large enough range.

[1] https://www.babylonjs.com/
[2] https://howlerjs.com/
[3] http://www.fresound.org

After ten preliminary trials, the results were so disappointing that we ended the experiments. The players who tried the game reported that it was frustrating and confusing trying to unlock the doors. We considered modifying the game to automatically open the doors after three failed attempts but instead decided to rethink our premise. At this point, it was apparent that lack of motivation was not the primary issue preventing users from accurately selecting the points. We decided to change our research question; instead of "Can we convey points in space using spatialized sound?", we set out to determine "How precisely can the average person select a position conveyed with spatialized audio?"

CHAPTER 3

DETERMINING LISTENER PRECISION

Our previous work showed us the need to determine the precision with which the average listener could select a point conveyed using spatial audio. An experiment was devised to help make that determination.

3.0.1 Methodology

Though many visualizations rely upon the mechanism of shape, there are other mechanisms that we could use with sound that could convey the same information denoted by shapes in visualizations. For instance, a particular shape type in a flowchart or UML diagram could be represented by a particular musical instrument in a sonification, as the shape itself is not important, but rather what it represents. Therefore, we reasoned that it would be more useful to determine if any perceptual commonalities existed between listeners using a system that conveys elements' position via the mechanism we used in our first study.

To that end, we revised our software from the shape presentation study to audibly present a point to a listener and then wait for the participant to touch the perceived position of that point on a touch screen monitor. The details of our setup can be found in [73, 74]. We conducted two studies with this setup; the first comprised single-point tests, while the second presented two points. Participants could replay the position as many times as needed and were allowed to replay audio stimuli representing the boundaries of the plane. Then they were able to select the position that they believed corresponded to the audio stimuli. The participants were shown the correct position after each trial.

Previously, we had relied on pure tones as audio stimuli in our work. Pure tones were adequate for our previous work; yet we felt it would be too confusing to convey multiple relationship types in one audio scene. We settled on the use of different timbres to convey the various types of relationship. Studies have shown that timbre discrimination in humans is very robust and sounds as short as a few milliseconds can be reliably identified [64]. After listening to a variety of audio samples, we settled on two that we felt were easily differentiable.

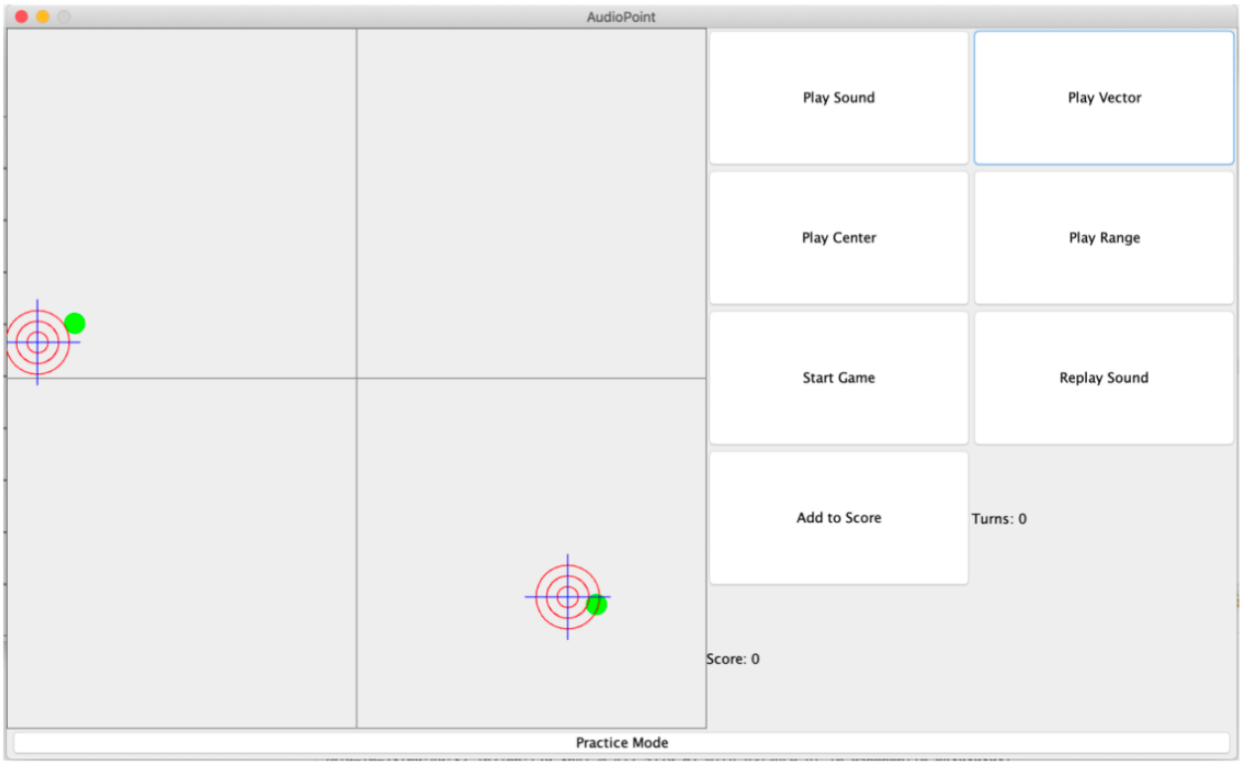

Figure 3.1 Sample point selection. The users' selected points are represented via concentric circle targets, and the software selected points represented via solid circles. Note that this is an example of the two-point test. The presentation plane measures 18.1 cm by 18.1 cm.

The first was a synthesized piano tone at 440Hz (Concert A or MIDI A4), while the second was a synthesized glockenspiel tone at 440Hz.

3.0.2 Results

We recruited undergraduate students, this time 29 (9 women, 20 men), with a mean age of $\mu = 20.3$ years ($\sigma = 1.5$ years). All reported normal hearing. Participants were allowed to practice until they reported that they were comfortable with the system. We then tested their precision. An analysis was performed to determine the mean precision (Figure 3.2) and found that the users were accurate to within 3.57 cm ($\sigma = 1.95$ cm). A regression analysis was performed to see if there was a correlation between practice trials (Figure 3.3) and overall accuracy, but found no significant correlation ($R^2 = 0.06$).

We decided to repeat the test and add a test that presented two points sequentially with a small delay. We again recruited students, finding 25 undergraduates (4 female, 21 male)

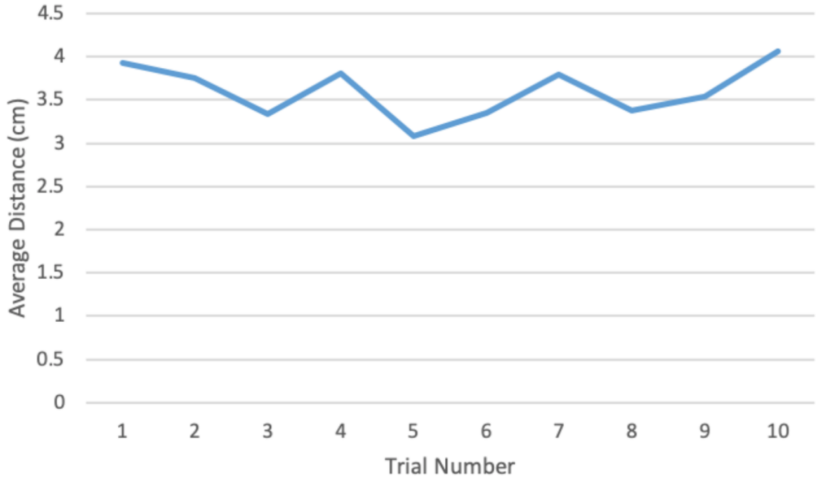

Figure 3.2 Mean accuracy per trial for the single-point only test. Accuracy did not increase with reinforcement. Originally published in [73].

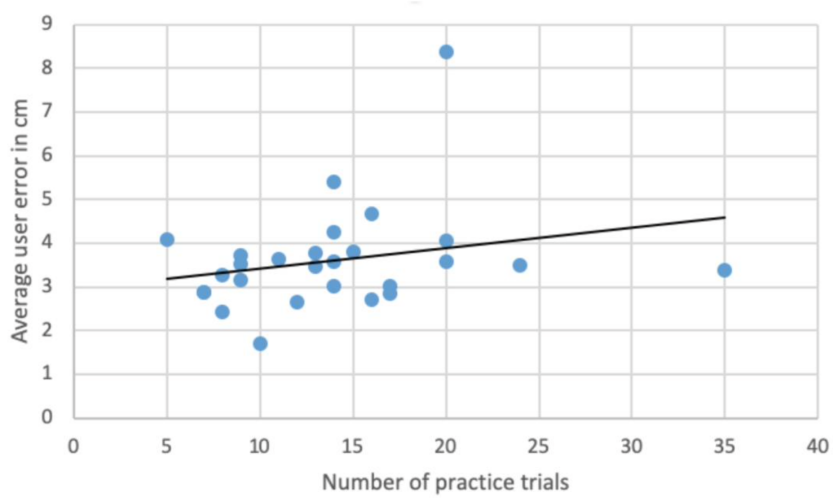

Figure 3.3 Number of practice trials and mean user error. Originally published in [73].

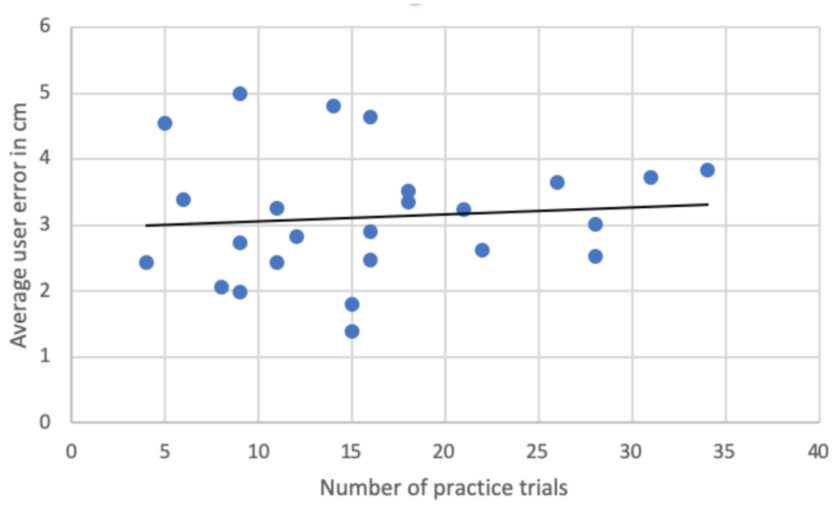

Figure 3.4 Number of practice trials and mean user error. Originally published in [73].

with a mean age of 21.8 years (σ = 4.0 years), all of whom reported normal hearing. For each participant, we first performed the same ten single-point tests and then ten trials with two points presented. The mean accuracy on the single-point task was 3.13 cm (σ = 2.24 cm), which closely mirrored the results of our first experiment. However, the two-point test showed poorer results, with a mean accuracy of within 4.33 cm for selecting the first point and a slightly lower accuracy of within 4.94 cm (σ = 3.23 cm) for selecting the second point. Analysis showed that the mean error in angle difference from the first and second points was 0.71 radians (40.68 degrees).

Again, there was no correlation between practice time and overall accuracy, with a regression value of $R^2 = 0.005$ (Figure 3.4). Five different intervals between the stimuli presented were tested in the two-point test to determine whether the delay between the stimuli affected the results. Figure 3.5 presents these results. A duration of one second yielded the best results, although the differences between the length and accuracy of the stimuli were minor.

3.0.3 Discussion

The results of this study indicate that with a reasonable division of the presentation space and the placement of graph elements in those divisions, users may be able to reliably

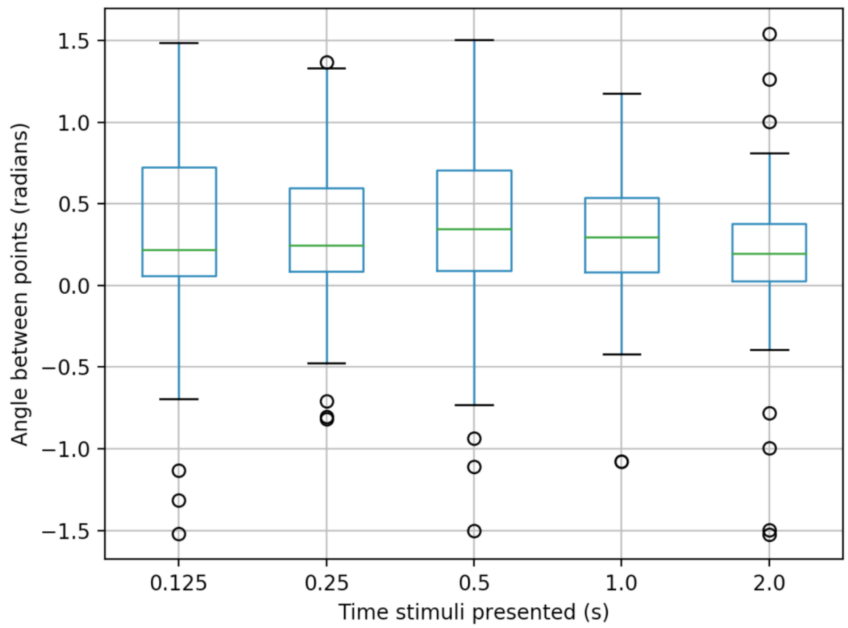

Figure 3.5 Boxplots of angle error compared to length of stimuli ($n = 250$). Originally published in [73].

perceive the position of an element relative to other elements.

We decided to modify our presentation of audio stimuli in future studies to address this issue. We found inspiration in a work from 2008 named iSonic that focused on the presentation of scatterplots to individuals with visual impairments [79]. iSonic presented a scatterplot via a nonet (a grid with nine squares). The scatterplot data were converted into a heatmap such that a higher data point density resulted in a higher value for a cell. Pitch-modulated audio tones represented the values of a particular grid cell, and users could interact with the system using a keyboard and a touchpad. Most importantly, the nonet structure was recursive; users could select a cell to "zoom" into it, resulting in the cell's expansion to the size of the entire grid. The presentation was then subdivided into nine cells, allowing users to explore the data in a very fine-grained manner (Figure 3.6). The iSonic authors tested the efficacy of their mechanism with a study comprising seven legally blind individuals (none reported having any residual vision). Individuals were asked to answer questions from three datasets using either the Excel or the iSonic system. The results showed that the

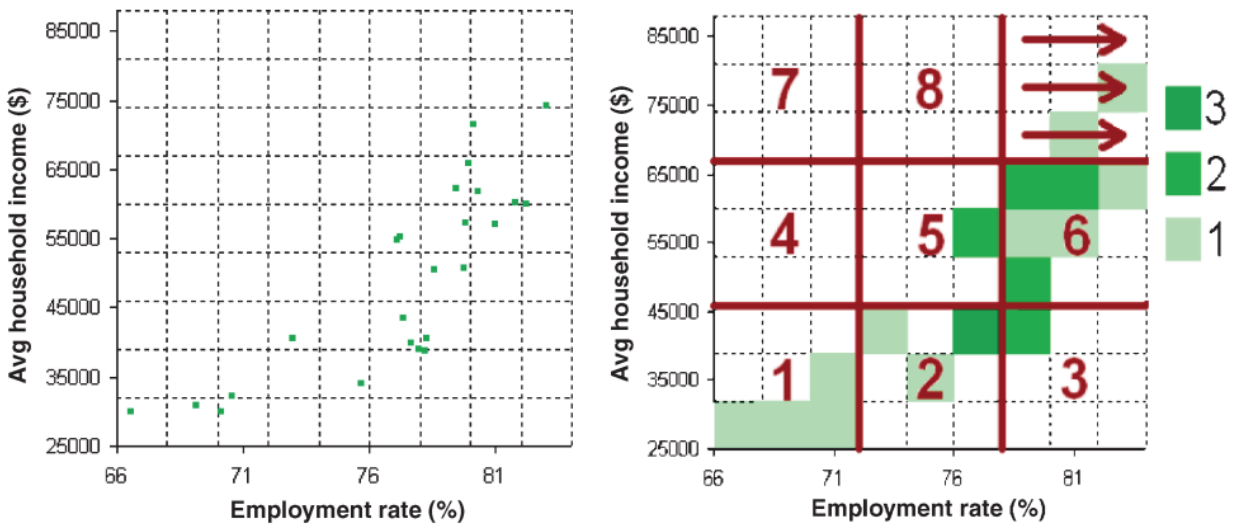

Figure 3.6 A scatterplot (left), and the same data presented using the iSonic software (from [79]. The software converts data point density to heatmap values. Users can zoom into a cell to have it expand to the entire nonet, which is then subdivided into cells for data exploration.

participants were able to answer questions as accurately with the iSonic system as they were with Excel (precision 86% in each); however, the participants reported that iSonic was easier to use than Excel (rating it 7.9/10 for ease of use compared to 7.0/10 for Excel).

The use of the nonet structure for exploration of data is appealing, for its simple design, recursive structure allowing for exploration of data at different levels, and because of our research on position and angle precision. We decided to adapt the nonet mechanism to the presentation of class diagrams. However, class diagrams differ greatly from scatterplots. Scatterplots represent a magnitude estimation task; for any given cell, stimuli must be provided that allow the consumer to compare the number of data points in that cell to other cells. However, class diagrams must convey the different types of relationships between diagram elements.

CHAPTER 4

USING NONETS AND SOUND TO CONVEY STATIC DIAGRAM ELEMENTS

Building on our previous work, we created software for a proof-of-concept. This time, we wanted to determine whether listeners could accurately recall static, structural information about classes using the nonet mechanism to divide our space and presenting the elements within the nonets using sound.

4.0.1 Method

Real-world class diagrams for large software projects can be very complex, while diagrams to convey educational concepts tend to be much smaller and simpler. For this reason, we decided to focus on the subset of diagrams most likely to be found in widely used software engineering texts. These diagrams illustrate concepts important to software engineering students, such as recurring design patterns, and usually contain a small number of classes. For example, the classic software engineering text *Design Patterns: Elements of Reusable Object-Oriented Software* contains twenty-three commonly used design patterns [34]. The sample structures provided by the authors of that text on average contained 4-5 classes ($n = 23, \mu = 4.5, min = 1, max = 10$). Figure 4.1 illustrates one of these educational examples, a very commonly used pattern called the Iterator pattern.

Zhao et al. showed that a *nonet*, or a grid with nine cells, can be used to convey graphical data via audio (Figure 3.6) [79]. Of particular importance is that each cell of their grid was

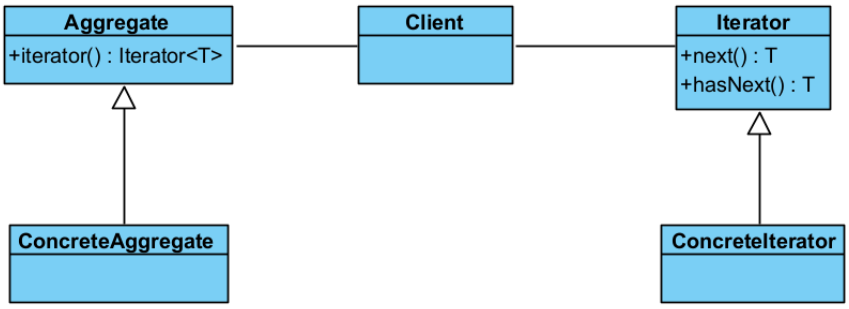

Figure 4.1 The Iterator design pattern is one of many commonly taught design patterns in introductory software engineering courses.

portrayed in a pre-defined order and that an empty cell was represented by either a pause or a short stimuli. Our work borrows the nonet concept. Given a list of classes, a user of our system may select a class from a hierarchy to examine more closely. The chosen class is placed in the center cell of the nonet. Classes that directly relate to the chosen class are placed around the outside of the nonet. When a user chooses to play a representation of the selected class and its relationships, the stimulus for each cell in the nonet (except the chosen class in the center) is played clockwise starting at the upper left cell and ending at the middle left cell. Any cell that does not contain a class directly related to the currently chosen class is conveyed by a short clicking sound.

Four relationship types may need to be portrayed in UML Class Diagrams. We will describe these in terms of two anonymously typed classes, class type `A` and class type `B`. An **association** occurs when class `A` holds a reference to one or more objects of type `B`. A **generalization** is a relationship whereby class `A` generalizes (as in is more generic than) class type `B`; these are used to illustrate parent-child or superclass-subclass relationships. A **realization** occurs when class `A` implements an interface specified by class type `B` (i.e. it implements the methods type `B` specifies). Finally, a **dependency** exists when class `A` requires class `B` to exist so that it can complete some task. Of these four relationship types, realization does not exist in all languages and can be simulated by generalization and abstract classes; therefore, it was not included in this proof-of-concept. Dependency relationships serve as a sort of "catch-all" type that can be complex to grasp for introductory students and were therefore also omitted from this work.

Audio stimuli were assigned to the two remaining relationship types; general MIDI instruments were used for simplicity. An association was represented by a half-second C4 note (MIDI note 60) with a synthesized acoustic grand piano (MIDI instrument 0). Generalizations were represented by a half-second C4 note with a synthesized glockenspiel (MIDI instrument 10). These were chosen by the researchers because of their disparate timbres. The Oracle Java JDK provided the synthesizer used for our MacOS system; however, the

sounds were replaced with the FluidSynth [1] sound font, as the default provided sound font was deemed of poor quality.

The participants listened to the stimuli via Sony MDR7506 Studio Monitor headphones connected to a four-channel Behringer Powerplay Pro-XL amplifier. As the amplifier was multichannel, the researchers were able to listen to the stimuli at the same time as the participants, and both the researcher and the participant had full control of the volume of their headphones. Participants could press a button on a gamepad to start the playback of the diagram. Each cell of the grid was played sequentially in clockwise order around the chosen class, which occupied the center of the grid. A short click was played if there was no class in a particular cell; otherwise, the association or generalization sound was played to indicate that a class in the current playing cell related to the chosen class via that type of relationship. Participants had a button that could repeat the diagram if they wanted, as well as buttons to control the length of the delay between stimuli. An illustration of our testbed layout is shown in Figure 4.2.

An invitation was extended to undergraduate students in computing courses at Grand Valley State University. The study population ultimately consisted of $n = 29$ undergraduate students (24 male, 5, female). The results of this study were originally published in [75]. The following questions were asked of the participants before participating in the study:

1. What is a class?
2. What is an instance variable?
3. What is inheritance?
4. What is polymorphism?
5. What is an association relationship in UML Class Diagrams?
6. What is a generalization relationship in UML Class Diagrams?
7. Which relationship type denotes that a class holds one or more instance variables of another class type?
8. Which relationship type denotes that a class is a parent or child of another class?

[1] https://www.fluidsynth.org/

Figure 4.2 Sample layout of a diagram using our testbed. `Client` was the chosen class and therefore occupies the central cell. Classes with relationships to this class are placed in the cells around the chosen class.

9. What is multiple inheritance?
10. What is a class hierarchy?

These questions were used to gauge whether participants understood the concepts of Object-Oriented Programming and UML Class Diagram well enough to participate. Participants who missed three or more of the questions were not invited to participate.

Those students who were unable to correctly answer all questions (but were still invited to participate) received a review of a few minutes. The participants were then shown the grid with numbers printed in the lower right corner of each cell. They were told how the cells in the diagram would be sequenced and asked to demo the sequence back to the researcher.

The researcher then described how a class diagram would be conveyed. The researcher then played a repeating sound while the participants put on the headphones and adjusted the volume to a comfortable level. The researcher then played with the stimuli used to represent an association, followed by the stimuli used to represent generalizations. When participants indicated that they were able to distinguish between the two and that they were

comfortable with the overall concept and had no questions, the researcher began to present sample diagrams. For this proof-of-concept, to ensure that a wide range of parameters were evaluated, the system randomly selected the number of associations and generalizations to present.

Participants were able to start the playback of a diagram when they were ready and were able to replay the diagram if they desired. The researcher then asked the participant to relate the number of associations and the number of generalizations that the diagram conveyed. Once participants indicated that they were comfortable with the sample diagrams, the researcher began keeping track of participant responses. Each participant was tested on ten diagrams.

For each diagram, the users were instructed that the central cell of the grid was representative of a class for which we wanted to know the number and types of relationships. The system generated a random number of related classes ($1 \leq n \leq 8$). Of these, between 0 and 2 were selected and assigned a generalization relationship to the class of focus. A small number was chosen for the possible number of generalizations, as it is uncommon (and often discouraged) to have classes inherit from more than one class. The remaining classes were assigned an association relationship to the class of focus.

After the test of ten diagrams, the researcher then asked the participants to listen to five additional diagrams. Participants were not asked to keep track of the number of associations and generalizations for these five tests, but rather for the users to keep track of the location (based on the grid number) where an association or generalization occurred. The purpose of the second test was to examine the efficacy of the nonet mechanism if also used in a top-down manner, as we believe that a complete system will require both methods.

This project was approved by the Institutional Review Board of the University.

4.0.2 Results

For the first test, all participants listened to ten diagrams and were asked to note the number of associations and generalizations for the focused class, generating $n = 580$ prompts.

The participants erred on the number of associations in a diagram a total of 11 times (3.8% error rate) and erred on the number of generalizations 14 times (4.8% error rate).

The types of errors that the participants made varied. The participants simply miscounted the number of associations, but counted the correct number of generalizations on $n = 2$ occasions (0.7% error rate). They miscounted the number of generalizations, but still gave the correct number of associations $n = 5$ times (error rate 1.7%. The most common error occurred when the participants miscounted both; this occurred in $n = 9$ of the trials (3.1% error rate). In all cases of a miscount for both categories, it appeared that the participants had mistaken one stimulus for the other, as in each instance a miscount in one category resulted in a miscount in the other category of the inverse amount. For example, a diagram may have conveyed 4 associations and 2 generalizations, but the participant perceived 5 associations and 1 generalizations, or 3 associations and 3 generalizations. The first type of error (perceiving a generalization as an association) occurred in $n = 7$ of these instances, while the second type (perceiving an association as a generalization) occurred in $n = 2$ of the instances.

The second test was harder for the participants. Each of the $n = 29$ participants had $n = 5$ trials to identify which cells of the grid contained a related class. The number of errors in all trials was 21 for an overall error rate of 14.5%. Two types of errors were made by participants; errors of omission when the participant did not perceive one of the stimuli ($n = 12$, or 57% of the errors), and errors occurring because the participant misidentified the cell ($n = 9$, or 43% of the errors).

4.0.3 Discussion

The results of the first test were promising. Participants were able to recall consistently, quickly, and accurately the number of associations and generalizations for a particular class. This is likely due to the low number of stimuli presented combined with a consistent and well-defined presentation space, i.e. the nonet design with a single class of focus at its center.

The results for the second test were less promising but should not be ignored. Although

participants erred at a rate of nearly 15% in this task, they received very little training time beforehand. It is very likely that familiarity with the system could improve these scores.

4.0.4 Future Work

The scope of this work was to design a proof-of-concept, and it was successful in that endeavor. Future work will be undertaken to create a UML browser that can load precreated UML diagrams or create diagrams from scratch or from existing software source code. User studies will then be conducted to determine whether students who explore UML diagrams via this system show similar levels of understanding as students who examine the same diagrams in a visual format.

The stimuli used to convey the types of relationship were chosen by the researchers in this experiment. A better option may be for participants to choose stimuli with perceptual meaning that are closely related to their preexisting schemas of relationship types [31].

Although our work focused solely on the presentation of the relationships between classes in a UML class diagram, this mechanism may be useful for the audible display of more general mathematical graphs. As mentioned in an earlier section, class diagrams do have similarities with mathematical graphs (node-edge graphs), which in turn have several similarities to other types of UML diagrams such as sequence diagrams, and even non-UML diagrams such as flowcharts.

In this study, no formal evaluation of the time taken by participants to answer questions after a diagram was presented (although we tracked the number of times a participant repeated a diagram). Future work should compare the time required for this mechanism with that required for visual UML. Additionally, while users noted the mechanism used for this study to be intuitive, we may want to formally evaluate the mental load using the NASA Task Load Index [2].

[2]https://humansystems.arc.nasa.gov/groups/tlx/

CHAPTER 5

REAL-WORLD DIAGRAM AND WORKLOAD TESTING

For this experiment we used the same hardware from the prior experiment, with the addition of a USB number pad. The system was prototyped in Python and then rewritten in Java. Java provided stability in production that the Python prototypes lacked, as Java's static type checking eliminated many run-time concerns. Visual Paradigm[1] was chosen as the software package for diagram creation, as it is widely used and readily available for free use by students at our institution through an academic partnership program.

Our first task was to understand the file format used by the Visual Paradigm software. Inspection of the files revealed them to be SQLite databases. SQLite is well-formed, and most languages provide libraries for the manipulation of these files. We used the Xerial SQLite JDBC driver [2]. Unfortunately, much of the data in these databases is unstructured; although there are a few tables, some are used for large text dumps. Thus, we had to use regular expression matching to filter out the data we required. In particular, Visual Paradigm provides the ability to export diagrams as XML files. However, we were unable to locate an XML schema to describe the data and help us parse the information needed from the files.

Once we completed the code that enabled us to pull the information we needed from the files, we wrote an application to allow users to select and explore diagrams from them. The application is designed to provide the often cited "Overview first, zoom and filter, then details-on-demand" model of information consumption [59]. To that end, we provide two views of the information. First, a diagram-level view is useful for discovering information, such as the number of classes in the diagram, their locations, and the areas with the least or most information. Second, is a detail-level view of a chosen class. The detail-level view provides information on the relationships between a selected class and the classes to which it is connected.

[1] https://www.visual-paradigm.com
[2] https://xerial.org/software/

Users are first presented with a list of the class diagrams within the file (there may be more than one). Once they select the desired diagram, our software presents that diagram with the diagram-level view. Upon presentation of the diagram, the software uses the Mycroft AI Mimic 3 text-to-speech engine[3] to vocalize the number of classes currently visible out of the total number of classes in the diagram. The diagram is positioned so that the upper left corner of the diagram is in the upper left corner of the presentation area. Entities in the diagram are presented in the same positions in which they were created in Visual Paradigm software.

For each view, there are two presentation modes that the user can select. These include verbose, text-to-speech-based methods and audio-only-based methods that rely upon the modulation of audio samples, but no speech, to convey diagram information. We designed the audio-based methods to allow users to interact more quickly with the system than is possible with speech.

Central to the audible presentation is the use of a nonet (a grid of nine cells), mapped to digit keys 1 through 9 on a USB number pad. The combination of a nonet and a number pad provides a convenient and intuitive correlation for input for exploration and navigation (Figure 5.1). This mechanism is used for both the diagram-level and detail-level views, though in subtly different ways, as described below.

5.0.1 The Diagram-Level View

The diagram-level view provides a top-down approach to exploring class diagrams. When a diagram is chosen from the loading screen, it is drawn with classes positioned exactly as they were placed in the Visual Paradigm software. The nonet is superimposed on the portion of the diagram that fits within the visible region of the software. The system is initially in verbose mode, and a spoken message alerts users to the number of classes currently in view and the total number of classes in the diagram. By pressing one of the digit keys on the number pad, a verbal message is played telling the number of classes in the corresponding

[3]https://github.com/MycroftAI/mimic1

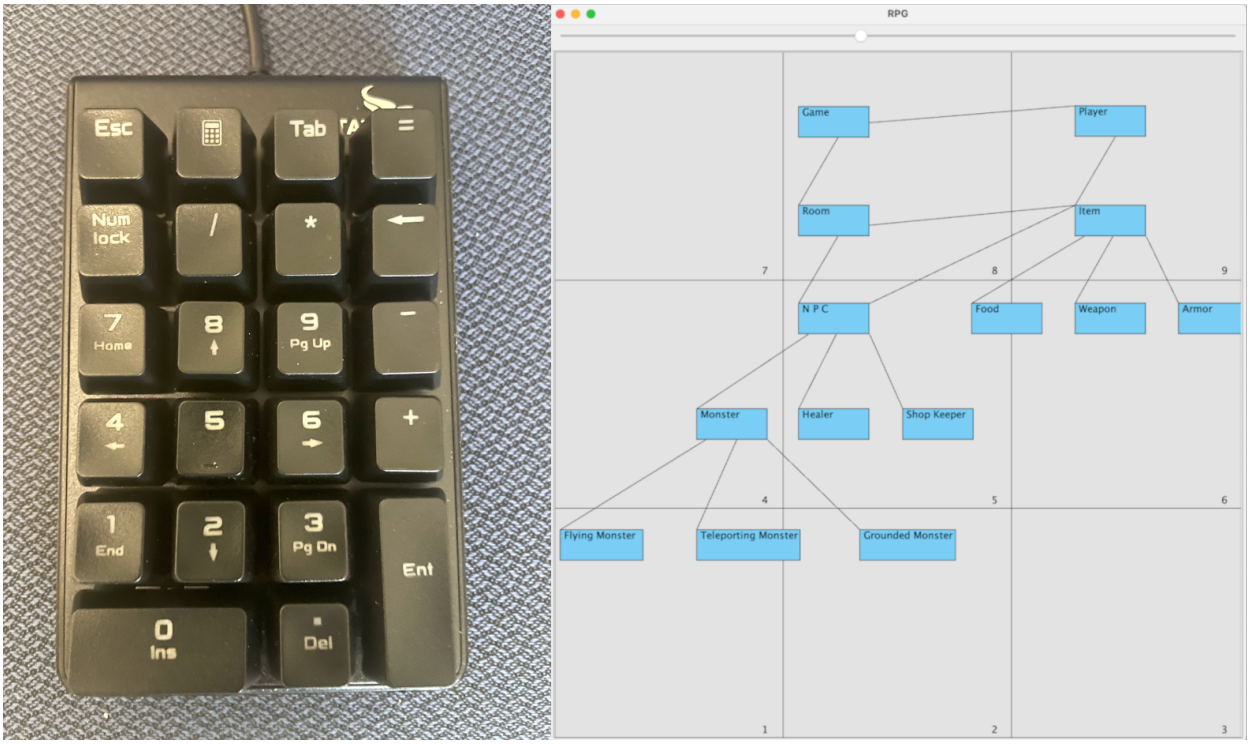

Figure 5.1 The system maps the digits 1-9 of the number pad to the cells of the nonet. Pressing a digit causes the system to render audio or speech (depending on the current mode) to describe the entities - if any - in that cell.

cell and a list of each of those class names. Pressing the period key causes the system to render the number of entities in each cell sequentially, left-to-right, top-to-bottom.

Although we designed the system for fairly simple, educational-use class diagrams, it is still possible that the entire diagram may not fit on the screen at once. Therefore, we programmed the system to pan the diagram using the W, A, S, and D keys (up, left, down, and right). Pressing one of these keys moves the diagram in the corresponding direction by the current width or height of a cell. When the diagram moves, an audio stimulus is best described as a short "whooshing" sound. The system will not allow one to pass past the boundaries of the diagram. Should the user attempt to move in a direction when they are already at the limit of that direction, a short "clicking" sound is played to alert them that the diagram has not moved.

The software allows users to zoom in on a particular cell using the "+" key to zoom in and the "-" key to zoom out. The zoom level is not arbitrary; rather, the system requires

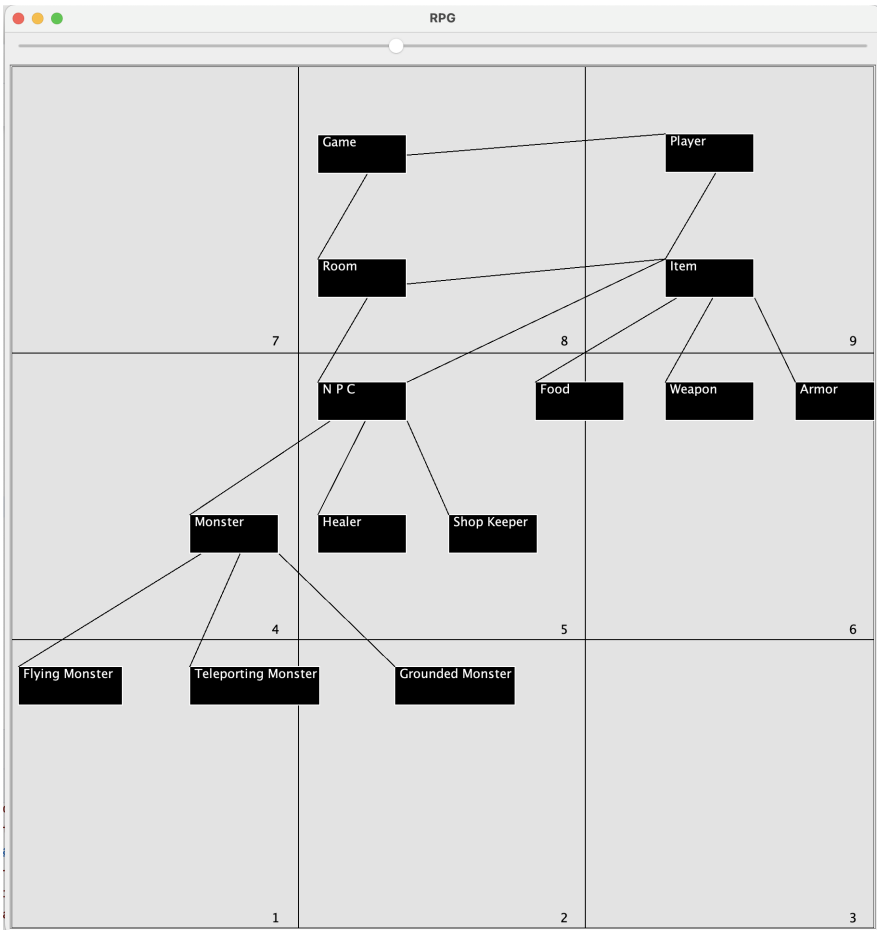

Figure 5.2 Sample diagram-level view. This view provides a high-level view of the diagram, with panning and zooming features. This view corresponds to the "Overview" phase of the "Overview, zoom and filter, then details" model of information consumption.

users to select a particular cell into which to zoom. The chosen cell is then expanded to the entire nonet. This recursive zoom mechanism is based on work by Zhao et al., described in the review of the literature in this paper [79]. When zooming into a cell, if there is only one class in that particular cell, the detail-level view is opened. Alternatively, the user can trigger a search for a class by pressing the "F" key. Doing so brings up a list of all the classes in the diagram, in alphabetical order. Selecting a class from this list automatically triggers the detail-level view for that class. Cancellation from this menu leaves the user in the current view. The list is programmed to speak the names of the classes in the list as the users move through them. Zooming out brings the user to the prior diagram-level view with the previous zoom level and panning applied.

Users may opt to switch out of verbose mode and into speechless audio mode by pressing the "E" key. In this mode, instead of presenting a voice that demonstrates the number of classes in a selected cell, the system plays a musical note with a tremolo effect applied. The amount of effect applied is governed by the number of entities in the selected cell. No tremolo is added if there is only one class in a cell. This provides users a rapid way to discover which areas of the map contain information and which may be ignored.

Unlike a traditional UML class diagram, the diagram-level view in the system does not present relationships between classes. The research team chose to separate the presentation of the relationships from the overall diagram, both because the previous work of the team had shown the need to limit the amount of audio stimuli and because research shows that the human auditory system is best suited for temporal input and vision is best suited for spatial input [36].

5.0.2 The Detail-level View

The detail-level view allows users to inspect a particular class and its relationships to other classes. This view is triggered whenever the user selects a class from the search menu or when zooming into a cell in which the class is the only diagram entity contained therein. This view also uses the nonet presentation mechanism but in a different way. In the detail view, the selected class is positioned in the center. Classes with which the selected class shares a relationship are positioned around the selected class, one class per cell. Although this provides only eight cells for related classes, this limitation is enough for the types of diagrams we wish to convey. Previous work identified the need for a mechanism for presenting diagrams with (on average) four to five classes [75].

Class diagrams represent relationships between classes with various line types, and directionality with different line-ending arrows. The audio corollary in this system is the use of stimuli with varying timbres combined with pitch modulation. For relationships, we use a waveform of a piano playing a C4 note to convey an association, and a waveform of a glockenspiel playing a C4 note for a generalization. Our corollary for the directionality is to

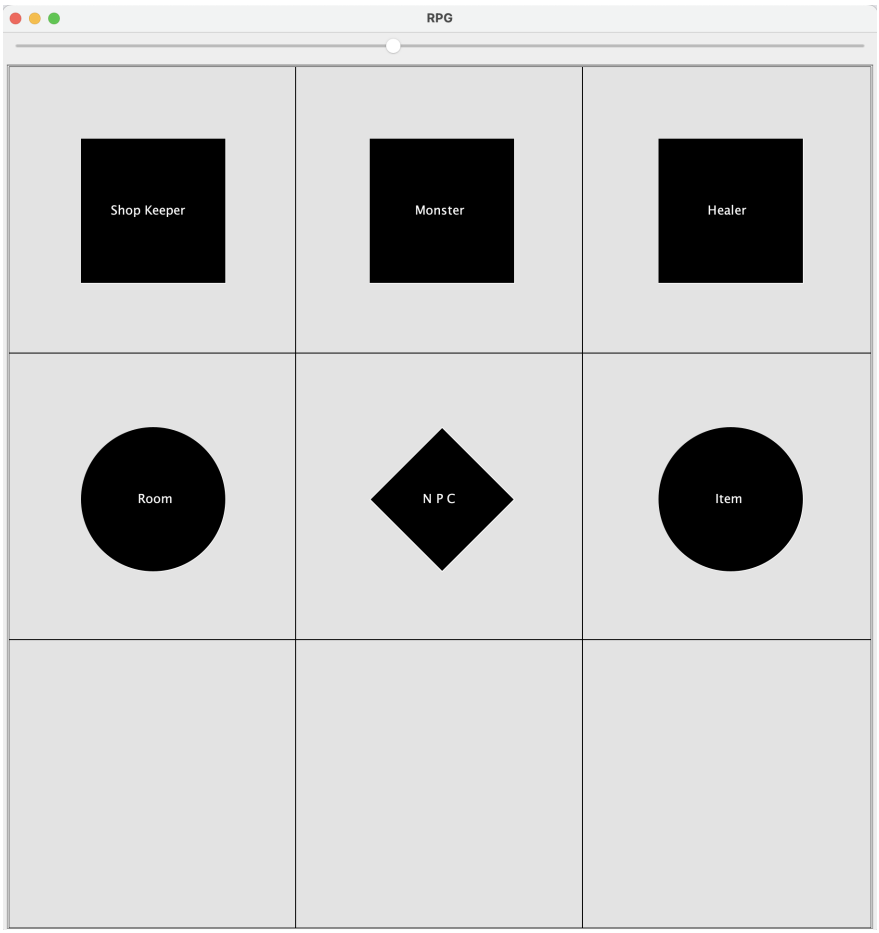

Figure 5.3 A sample detail-level view. The center class (NPC in this case) is the class selected through zooming deeper into the diagram-level view. The different types of relationships are represented with different shapes for those with low-vision, but with different audio stimuli for individuals with blindness.

modulate the waveform for the relationship such that a direction away from the currently selected class rises in pitch, and a direction toward the current class is a sinking pitch. The pitch change is applied to the last half-second of the stimuli, with a change in pitch of a whole step. When in verbose mode, the name of the class is spoken after the audio stimuli is played.

5.0.3 Experiment

We asked for volunteers from undergraduate computing students. All students were required to have completed computer science one and two at the university where we teach. We imposed this requirement, as our university teaches class diagrams in Computer Science

Two; therefore, all participants should have been exposed to them prior. We wished to compare performance of three groups; a group using traditional visual UML class diagrams, one using the Visual Paradigm software with a screen reader, and one with our system. However, we tested Visual Paradigm software with Apple's Voice Over, Microsoft Window's Narrator, and Freedom Scientific's JAWS screen reader, and found that none of them was able to render any diagram information. The menus and menu items were described only as "System dialog", or a message alerting us that we were in a "JScrollPane". It was impossible to use a screen reader with Visual Paradigm. We therefore recruited only two groups to test our software. Both groups consisted of 30 undergraduate students.

Each participant worked individually with a researcher. For both groups, participants were first asked if they remembered UML class diagrams from their Computer Science Two course. We then asked them the preliminary questions from Section 4 that cover basic Object-Oriented Programming concepts and definitions, as well as questions specific to class diagrams. When students struggled to correctly answer a question, we reviewed the material with them. We were unable to find students with visual impairments or blindness. Although this limitation is unfortunate, it provided us with the opportunity to review the class diagrams visually with each participant before having them answer questions about the diagrams used in the study.

The participants in the first group were then blindfolded. We described our system to them using the script from Appendix B. Participants were allowed to ask questions and then given an open-ended period to use the system and grow familiar with it, while still asking questions of the researchers. During this time, they were given a sample diagram to explore. Once a participant indicated they were ready to begin, we loaded a new diagram and began the experiment.

Each participant was asked to answer ten questions about the diagram presented to them. Once a question was asked, the researcher began a timer on a stopwatch. When a question was answered or participants declined to answer the question, the timer was stopped, and the

time was recorded. The questions were asked one at a time and participants were given as much time to answer as they wished. During this time, participants could ask for clarification about the stimuli or how the system worked, but were not allowed to ask specific questions about the diagrams or about UML. All of the participant's interactions with the system, as well as the timer information and answers, were recorded in log files for later data analysis.

The second group was given the same opportunity to review Object-Oriented Programming and UML class diagrams, before being tested on a traditional, visual, UML class diagram. Participants were asked questions one at a time and given as much time as needed to answer (or declined to answer if they could not determine an answer). The researchers tracked the time used to reach an answer with a stopwatch and recorded the times and answers in log files.

After completing the ten basic structural questions, each participant was asked to complete a NASA Task Load Index (NASA TLX) to provide information on the workload of answering the questions with the given presentation mechanism. Finally, participants were asked to answer three higher-level thinking questions about their diagrams. During this time they were not allowed to refer back to the diagrams but were asked to answer the questions from memory. This was to provide some measure of the participants' synthesis and understanding of the diagrams they had encountered.

5.0.4 Results

The first group consisted of 30 undergraduate computing students (19 male, 11 female). As noted above, participants were given as much time as they wanted to practice with the system prior to testing. On average, the participants practiced for 930.79 seconds ($\sigma = 232.62$ seconds), or around 15 and a half minutes. The minimum time taken by a participant was 635.92 seconds (10 minutes, 35.92 seconds), and the longest was 1370.94 seconds (22 minutes, 50.94 seconds). A Pearson correlation coefficient was calculated between practice time and the number of errors made by the participants, but found that the factor was not strongly correlated with success ($r = 0.19$).

	1	2	3	4	5	6	7	8	9	10
Audio Presentation	1.0	0.93	0.87	0.97	0.93	0.97	1.0	0.97	0.97	0.63
Visual Presentation	1.0	1.0	1.0	0.97	1.0	1.0	1.0	1.0	1.0	0.90

Table 5.1 Difficulty index values for each question, separated by presentation mode. Lower values indicate harder difficulty questions.

The second group comprised a different set of 30 undergraduate students (22 male, 8 female). This group served as the control group and was thus given the traditional visual UML task. Participants were asked the same questions and had the same review time as those in the audio group, but did not practice answering UML questions before beginning the experiment.

For each question, we calculated a difficulty index so that we could identify the questions and the types of questions the students struggled with. This also served to compare the results of the two presentation methods. The questions were written to be simple structural questions for students to answer, given their prior experience with class diagrams. We wanted to ensure that the errors made were most likely due to the presentation mechanism and not to the complexity of the question. We used the function

$$P = \frac{\text{Number of correct responses}}{\text{Total number of responses}} \tag{5.1}$$

to generate the index values. The values of the questions separated by presentation type are in Table 5.1. We considered a question to be difficult for index values < 0.80. Only one question, number 10 presented with audio, was flagged as a hard question. Of importance is that the same question presented visually was not flagged as hard.

The amount of time needed to answer a question varied greatly due to the presentation

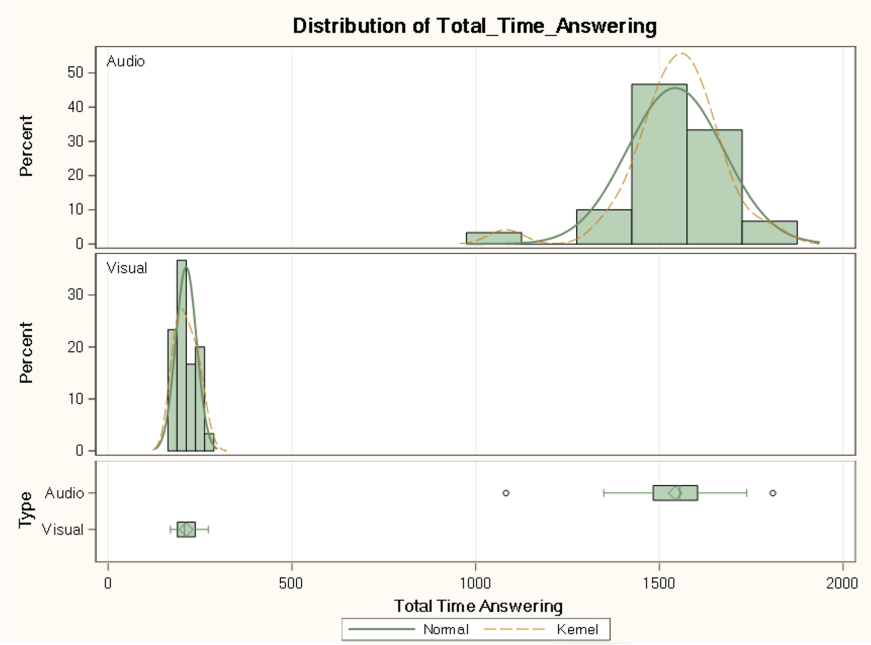

Figure 5.4 Distribution of response times. As expected, audio task response times were significantly greater than the visual task.

format. The average time taken to answer a question using audio-based presentations was 154.33 seconds ($\sigma = 61.05$ seconds), while the time to answer for visual-based presentations was 21.22 seconds ($\sigma = 10.11$ seconds). The response time distributions are shown in Figure 5.4. In addition to the difference in response times, it is notable that the audio task resulted in more outliers (both of a shorter and a longer response time). This could indicate that some of the participants were more comfortable with the task than others, which could indicate that more training would be helpful. The minimum time to respond to a response was 12.82 seconds for the audio-based method and 4.24 seconds for the visual method. The maximum response time was 325.43 seconds for the audio-based method and 62.62 seconds for the visual method. The mean time-to-answer values are shown in Figure 5.5.

We conducted a Two-sampled T-Test to determine if there were significant differences in response times. As time variances differed between the two samples, we used the Satterwthwaite test. This showed that the time variations differ significantly ($p < 0.0001$). This is to be expected, as the audio task uses a sequential presentation method, whereas the visual presents all information at once.

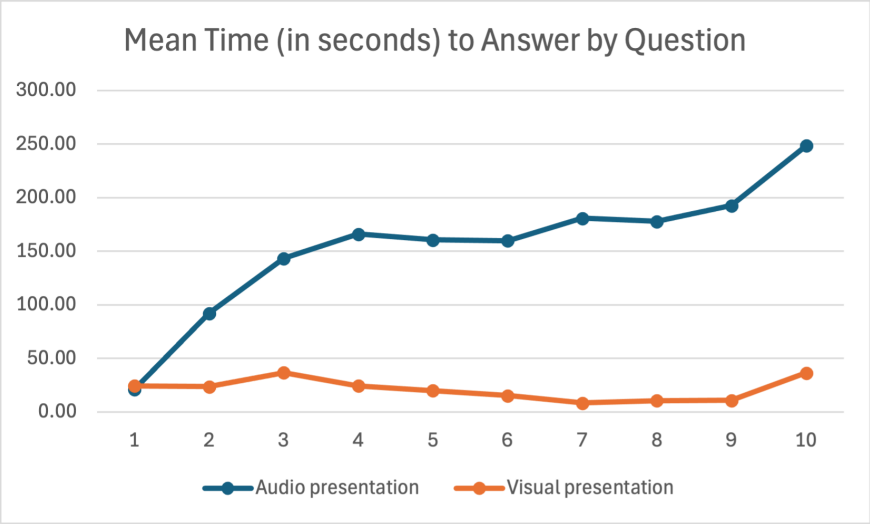

Figure 5.5 The mean time in seconds, from the time a question was asked until the participant answered, per question.

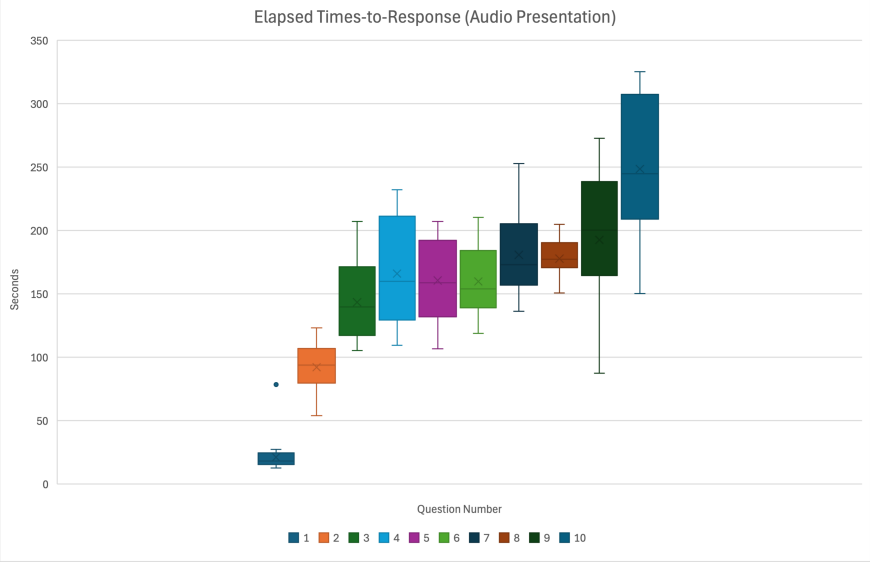

Figure 5.6 Time variance of responses to each question, under the audio-presentation approach.

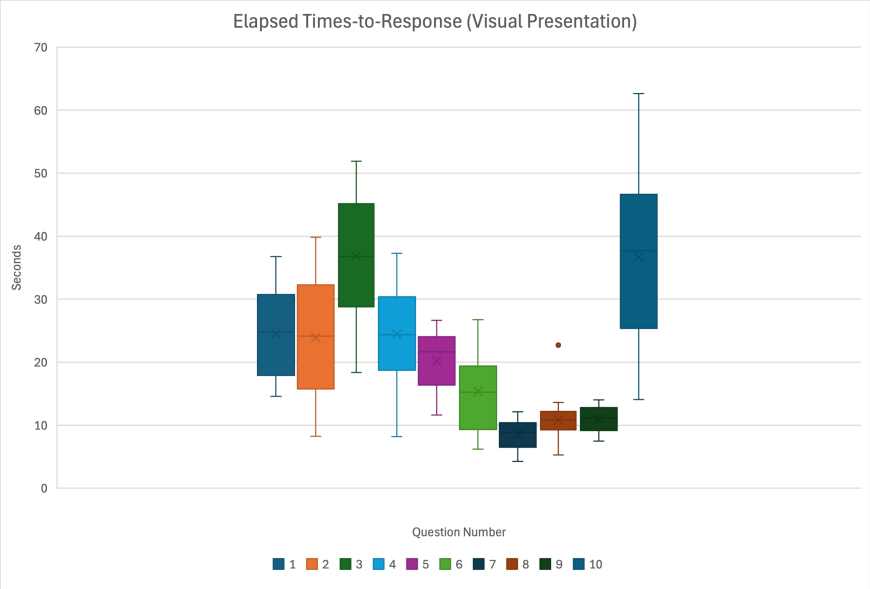

Figure 5.7 Time variance of responses to each question, under the visual-presentation approach.

5.0.5 Workload Comparison

In order to determine the perceived workload level of using the audio-based system, participants were asked to complete a NASA Task-Load Index (TLX) after completing a test. This index is highly regarded and used in many different fields and studies to help designers understand the subjective experience of various types of workload for users of systems [37]. The index compares the scores in the six areas of mental demand, physical demand, temporal demand, performance, effort, and frustration (Table 5.2). Users select a value for their perceived workload for each of the types. They also go through a pairwise selection process, where they rank each of the workload types against each other to determine which factors should be weighted the highest when calculating an overall index score. The final workload scores are between 0 and 100. We used the Apple iOS version of the app on an Apple iPad Air to collect the data.

The task load scores for the audio presentation tasks had a mean of 22.76 ($\sigma = 10.22$). Although this represents a low-to-medium workload overall, we focussed on subscale scores to get a better idea of the complexities users encountered when using the mechanism. The scores

Mental Demand (low/high) - How much mental and perceptual activity was required?
Physical Demand (low/high) - How much physical activity was required?
Temporal Demand (low/high) - How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred?
Performance (good/poor) - How successful do you think you were in accomplishing the goals of the task set by the experimenter?
Effort (low/high) - How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration Level (low/high) - How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent were you?

Table 5.2 Definitions of each workload type. From the NASA-TLX iOS app.

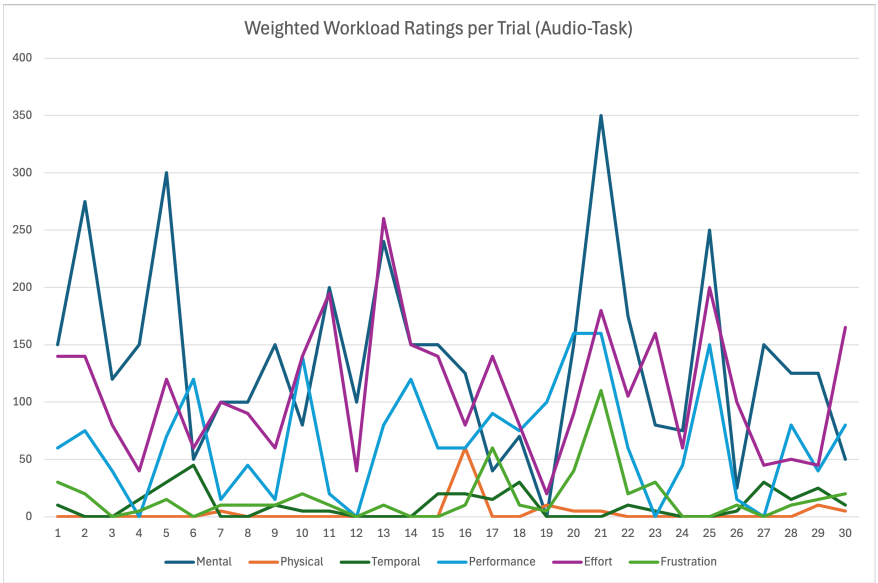

Figure 5.8 Weighted rankings of workload measures, per trial. Mental demand had the highest workload values, followed by effort.

were scaled by how participants completed the pairwise comparison tasks, which allows users to assign importance to individual workload tasks. Mental demand had the highest scaled score, with an average of 136.83 (σ = 82.31), followed by effort (μ = 109.17, σ = 57.42), performance (μ = 65.83, σ = 48.51), frustration (μ = 16.00, σ = 22.30), temporal demand (μ = 10.17, σ = 12.07), and physical demand (μ = 3.33, σ = 11.09). Workload rankings per trial are in Figure 5.8.

The mean value of the workload for the visual task was μ = 14.38 (σ = 10.22). Again, we examine subscale scores to better understand user perception of the task. For the visual

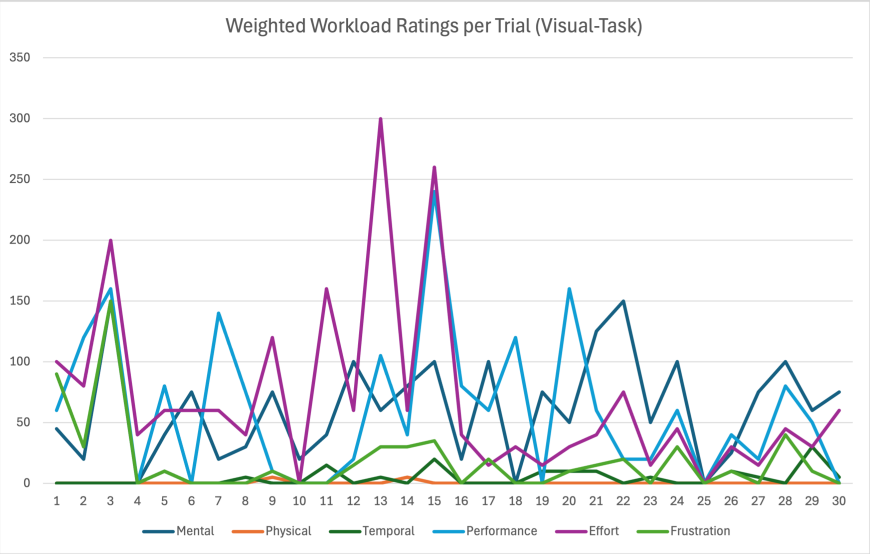

Figure 5.9 Weighted rankings of workload measures, per trial for the visual task. Effort showed the highest workload value, followed by mental demand.

task, effort was the highest rated workload with $\mu = 69.50$ ($\sigma = 72.22$), followed by mental workload ($\mu = 62.00$, $\sigma = 41.68$), performance ($\mu = 60.67$, $\sigma = 60.01$), frustration ($\mu = 18.50$, $\sigma = 31.35$), temporal workload ($\mu = 4.67$, $\sigma = 7.18$), and physical workload ($\mu = 0.33$, $\sigma = 1.27$).

To determine whether a particular workload factor was significantly different for the audio task compared to the visual task, we were unable to use a standard T-test, as the underlying data was not continuous and the samples did not have similar variance or adhere to a normal distribution. Therefore, we used a Wilcoxon rank sum test. These nonparametric tests use a data ranking method that allows us to compare the median values and the distribution of samples (Figure 5.10).

The scores indicate that the workload values for mental workload and effort are significantly different ($p < 0.0001$ and $p = 0.0015$, respectively). The temporal workload values are close to statistical significance, but at $p = 0.0705$ they do not meet the threshold of $p < 0.05$. Examining the distributions of the rank scores via histograms reinforces these results. We see in Figure 5.11 that the distribution of the data for mental workload is higher for the audio task than for the visual task.

Measure	Type	Minimum	Median	Maximum	Mean-Score	P-Value
Mental	Audio	0	125	350	39.45	<.0001
	Visual	0	60	150	21.55	
Physical	Audio	0	0	60	33.1	0.0773
	Visual	0	0	5	27.9	
Temporal	Audio	0	5	45	34.33	0.0705
	Visual	0	0	30	26.67	
Performance	Audio	0	60	160	32.1	0.4839
	Visual	0	55	240	28.9	
Effort	Audio	20	100	260	37.85	0.0015
	Visual	0	45	300	23.15	
Frustration	Audio	0	10	110	31.23	0.7424
	Visual	0	10	150	29.77	

Figure 5.10 Wilcoxon Rank-Sum score summaries comparing the workload values of the different presentation methods. P-value scores indicate that mental workload and effort showed significant differences, with the audio method being higher.

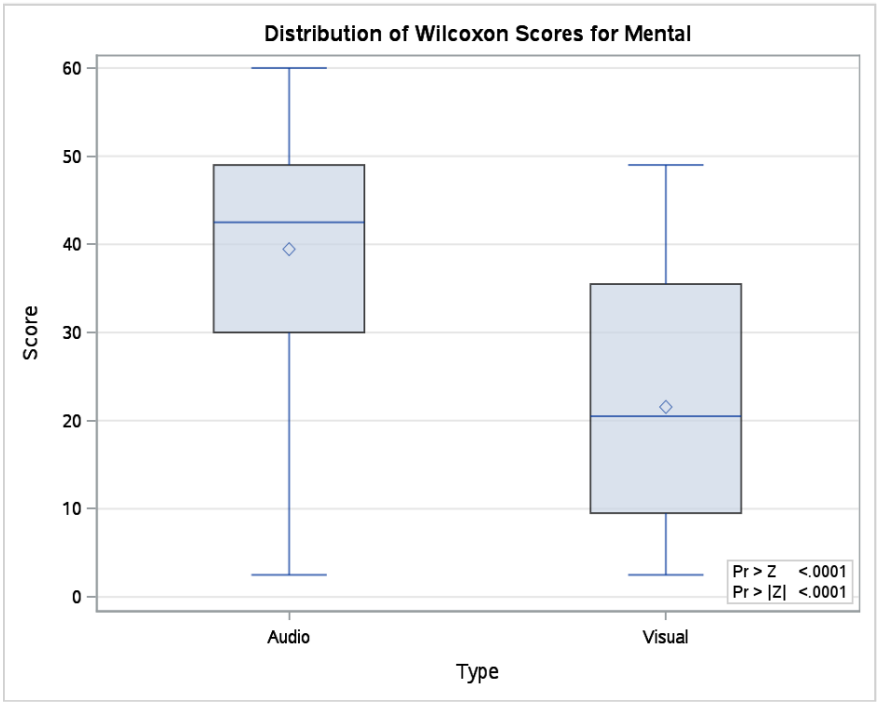

Figure 5.11 The distribution for Wilcoxon rank-sum scores for mental workload. The two methods show clear differences in distribution, with the audio presentation mechanism resulting in higher values. This indicates participants felt the mental workload was harder for the audio based task.

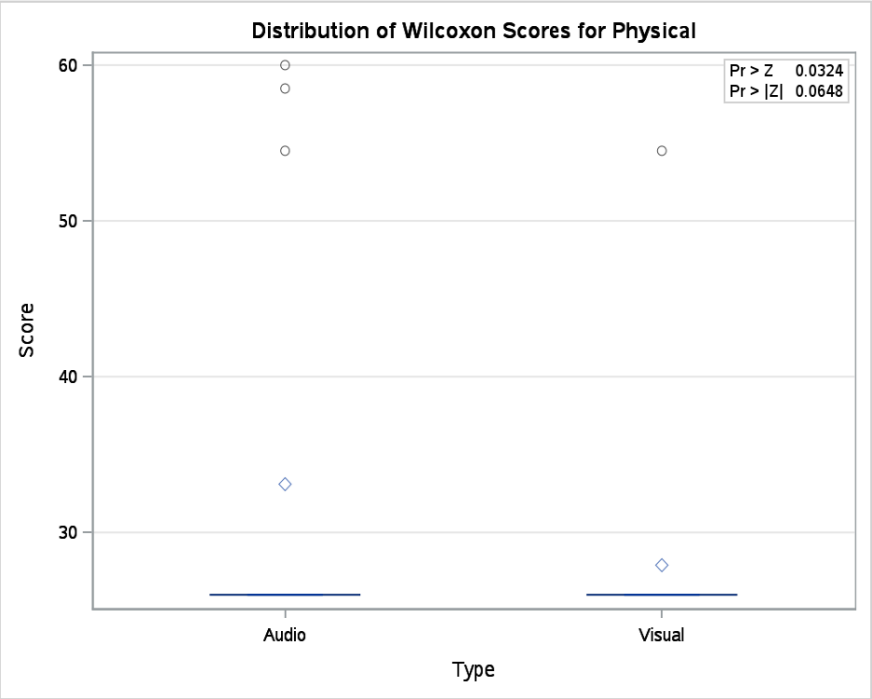

Figure 5.12 The distribution of Wilcoxon rank-sum scores for physical workload. The different presentation mechanisms did not result in significant differences in these values, though the audio method yielded a greater number of outliers.

Physical, temporal and performance workload scores (Figures 5.12, 5.13, and 5.14) did not differ significantly. Interestingly, though participants for both audio and visual tasks were specifically told that they had as much time as needed to answer the questions, both reported feeling time pressure. We are unsure what to make of this result.

Effort scores (Figure 5.15) varied significantly. Again, this is to be expected. However, we were surprised to find that the frustration scores were remarkably similar. This was an unexpected finding; we had hypothesized that students would find the audio task more frustrating. This could be a reflection of the general discomfort of students with UML diagramming.

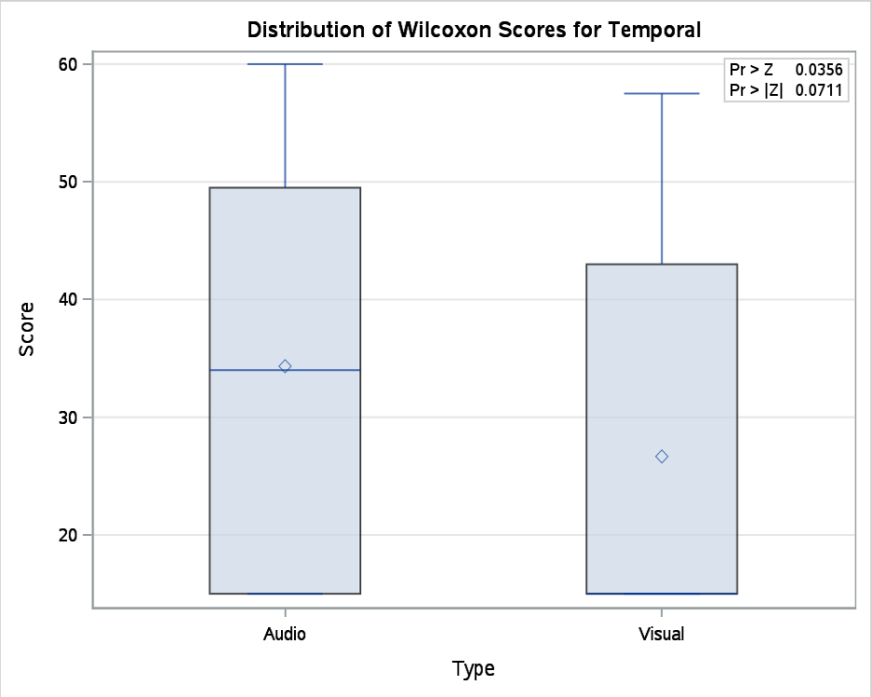

Figure 5.13 The distribution of rank-sum scores for the temporal task. Oddly, though both groups were given as much time as needed to answer questions, these scores indicate they felt time pressure.

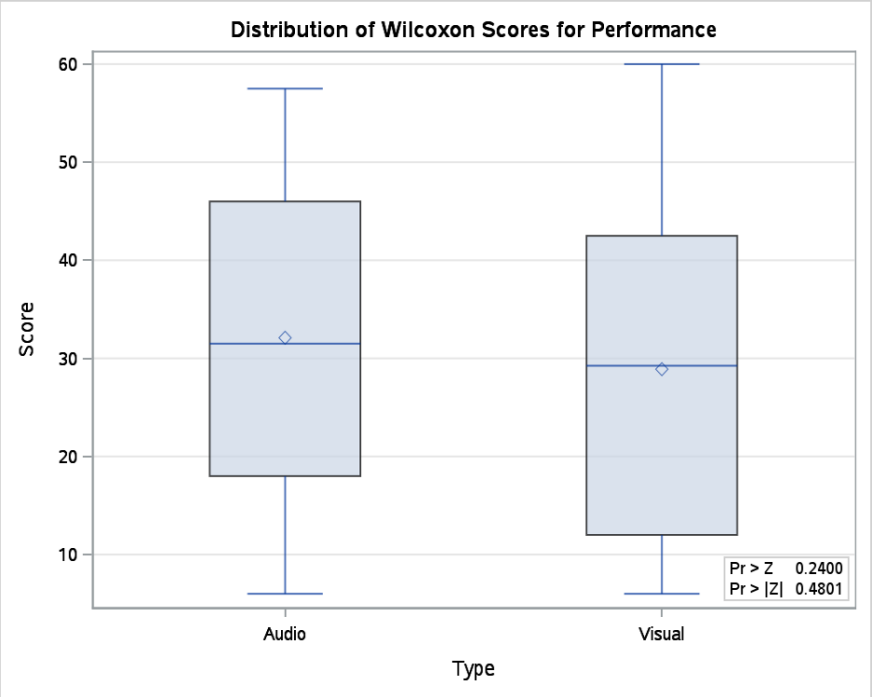

Figure 5.14 Performance rank-sum scores were very similar between the two tasks. Even though students were not being graded on the task, they reported similar measures of feelings of success with the tasks.

Figure 5.15 Rank-sum scores for effort were significantly different between the tasks. This indicates that the perceived effort for successfully completing the audio task was higher than the visual one.

Figure 5.16 Frustration scores between the two tasks were very similar. This is a surprising result, and likely indicates general student struggles with UML diagramming.

CHAPTER 6

CONCLUSIONS

This work demonstrates a method for conveying UML class diagrams using audio. Although this work began with a strong focus on using spatial audio to convey the relative positions of elements in class diagrams, the results of multiple experiments led us to the realization that the precision with which humans perceive positions of elements through spatial audio alone is insufficient for this task. Relationships between elements in class diagrams can be better conveyed through the use of a well-defined presentation space, the nonet, or grid of nine cells, combined with multiple views of the data (both using the nonet for layout) and modulated audio of differing timbres. Spatial audio is still utilized in the detail-level view; however, its use in the view is only of supplemental importance, as a fixed sequence of cells in the nonet is the primary mechanism used to convey relative position.

It is unlikely that division of the presentation space into more cells could lead to better results, as our previous work showed that the precision of listener perception lacked fidelity. Cells in a nonet structure are localized, and provide a set of cells that is a small enough number that listeners can readily keep track of relationships. The ability to pan and zoom in on a cell compensates for the small number of cells. Additionally, the ability to directly map the nonet to a common numeric keypad provides an inexpensive and readily available interface.

Although the mental workload of this method is slightly higher than that of traditional visual class diagram presentation, the difference is relatively small. Undoubtedly, the addition of the remaining element types, such as interfaces and dependency and realization relationships, may increase the workload; however, with careful choice of psychoacoustic properties combined with filtering and zooming capabilities, any increase in the workload can likely be controlled. This is likely the same for the effort workload.

Of interest is that the participants reported that answering the questions using traditional visual class diagrams *was just as frustrating as using our audio-based method.* This was

a surprising result believed to be due to general student discomfort with class diagrams. Combined with the results of the UML struggle survey, we believe that this is largely due to inconsistent use throughout the curriculum. It is highly likely that more consistent use could reduce this source of frustration.

In particular, this work shows promise in effectively conveying those class diagrams types commonly used in educational settings. Diagrams in the educational setting are typically comprised of a relatively small number of elements and element types, and exist primarily to teach a software design pattern or concept. The results show that this work is suitable for that purpose.

6.0.1 Future Work

Future work may focus on the addition of those less-used element types and relationships, such as interfaces, dependencies, and realizations. Furthermore, an examination of alternate stimuli should be performed, as it is likely that the workload can be further reduced by increasing the disparity between stimuli. Additionally, a more robust solution would likely allow for the customization of the stimuli so that users can adapt the mechanism to better suit their mental schemas.

Our mechanism focuses on the presentation of diagrams with elements positioned exactly as in the original graphs. It is possible that modifying the layouts of the class diagrams while still preserving the semantic meaning might lead to better results, though this could complicate communication about diagrams between individuals and teams. However, some work has shown that the quality of the layout of a class diagram can affect the understanding and comprehension of diagrams [62]. This factor may be particularly important when portraying diagrams to individuals with disabilities and thus merits further study.

This approach has not been tested with large and dense diagrams such as those that represent larger software engineering projects. Ultimately, it would be most beneficial to the community that this work be useful for diagrams of any size and for diagrams beyond UML.

BIBLIOGRAPHY

[1] Brown v. Board of Education of Topeka, 347 U.S. 483.

[2] Web content accessibility guidelines (wcag) 2.1. https://www.w3.org/TR/WCAG21/.

[3] Pennsylvania Association, of Retarded Children v. Commonwealth of Pa., 1972.

[4] The National Curriculum in England - GOV.UK, Dec 2014.

[5] Fact sheet: President obama announces computer science for all initiative 2016. FACT SHEET: President Obama Announces Computer Science For All Initiative, Jan 2016.

[6] Accessibility principles. https://www.w3.org/WAI/fundamentals/accessibility-principles/, May 2019.

[7] Students with disabilities. https://nces.ed.gov/programs/coe/indicator/cgg, May 2021.

[8] Pypl popularity of programming language. https://pypl.github.io/PYPL.html, Sep 2023.

[9] Tiobe index for october 2023. https://www.tiobe.com/tiobe-index/, October 2023.

[10] Iyad Abu Doush, Enrico Pontelli, Dominic Simon, Tran Cao Son, and Ou Ma. Making Microsoft Excel™: multimodal presentation of charts. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, pages 147–154, 2009.

[11] Iyad Abu Doush, Enrico Pontelli, Tran Cao Son, Dominic Simon, and Ou Ma. Multimodal presentation of two-dimensional charts: an investigation using open office xml and microsoft excel. *ACM Transactions on Accessible Computing (TACCESS)*, 3(2):1–50, 2010.

[12] Jürgen Anke, Stefan Bente, V Thurner, O Radfelder, and K Vosseberg. Uml in der hochschullehre: Eine kritische reflexion. In *CEUR Workshop Proceedings*, pages 8–20, 2019.

[13] Fred Attneave and Richard K Olson. Pitch as a medium: A new approach to psychophysical scaling. *The American journal of psychology*, pages 147–166, 1971.

[14] Albert Bergman. *Auditory scene analysis the perceptual organization of sound*. MIT Press, Cambridge, Mass., 1990.

[15] Jeffrey P. Bigham, Ryan S. Kaminsky, Richard E. Ladner, Oscar M. Danielsson, and Gordon L. Hempton. Webinsight: Making web images accessible. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '06, page 181–188, New York, NY, USA, 2006. Association for Computing Machinery.

[16] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley Longman, Inc., 1999.

[17] Robert G Brookshire. Teaching uml database modeling to visually impaired students. *Issues in Information Systems*, 7(1):98–101, 2006.

[18] Matt Calder, Robert F. Cohen, Jessica Lanzoni, Neal Landry, and Joelle Skaff. Teaching data structures to students who are blind. *SIGCSE Bull.*, 39(3):87–90, jun 2007.

[19] Sarah Carruthers, Amber Thomas, Liam Kaufman-Willis, and Aaron Wang. Growing an accessible and inclusive systems design course with plantuml. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 249–255, 2023.

[20] Stephen Cass. The top programming languages 2023, Aug 2023.

[21] E Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the acoustical society of America*, 25(5):975–979, 1953.

[22] Stanislav Chren, Barbora Buhnova, Martin Macak, Lukas Daubner, and Bruno Rossi. Mistakes in uml diagrams: analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109. IEEE, 2019.

[23] Florian Daniel and Maristella Matera. *Model-Driven Software Development*, pages 71–93. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[24] Mohamed Debashi and Paul Vickers. Sonification of network traffic flow for monitoring and situational awareness. *PloS one*, 13(4):e0195948, 2018.

[25] Alfonso Delgado-Bonal and Javier Martín-Torres. Human vision is determined based on information theory. *Scientific reports*, 6(1):36038, 2016.

[26] Alexandra Diehl, Alfie Abdul-Rahman, Mennatallah El-Assady, Benjamin Bach, Daniel A Keim, and Min Chen. Visguides: A forum for discussing visualization guidelines. *EuroVis (Short Papers)*, 6(7), 2018.

[27] Brad Doherty and Betty HC Cheng. Uml modeling for visually-impaired persons. In *HuFaMo@ MoDELS*, pages 4–10, 2015.

[28] Gaël Dubus and Roberto Bresin. A systematic review of mapping strategies for the sonification of physical quantities. *PloS one*, 8(12):e82491, 2013.

[29] Mohamad Eid, Atif Alamri, Jamil Melhem, and Abdulmotaleb El Saddik. Evalutation of uml case tool with haptics. In *Proceedings of the 2008 Ambi-Sys workshop on Haptic user interfaces in ambient media systems*, pages 1–5, 2008.

[30] Jamie Ferguson and Stephen A. Brewster. Evaluation of psychoacoustic sound parameters for sonification. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ICMI '17, page 120–127, New York, NY, USA, 2017. Association for Computing Machinery.

[31] Jamie Ferguson and Stephen A Brewster. Investigating perceptual congruence between data and display dimensions in sonification. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–9, 2018.

[32] Jamie Ferguson and Stephen A. Brewster. *Investigating Perceptual Congruence between Data and Display Dimensions in Sonification*, page 1–9. Association for Computing Machinery, New York, NY, USA, 2018.

[33] John M. Findlay. Active vision [electronic resource] : the psychology of looking and seeing / john m. findlay and iain d. gilchrist., Jan 2003.

[34] Erich Gamma. Design patterns: elements of reusable object-oriented software, 1995.

[35] Corentin Guezenoc and Renaud Seguier. A wide dataset of ear shapes and pinna-related transfer functions generated by random ear drawings. *The Journal of the Acoustical Society of America*, 147(6):4087–4096, 2020.

[36] Sharon E Guttman, Lee A Gilroy, and Randolph Blake. Hearing what the eyes see: auditory encoding of visual temporal sequences. *Psychol Sci*, 16(3):228–235, March 2005.

[37] Sandra G. Hart. Nasa-task load index (nasa-tlx); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.

[38] Ed Júnior, Kleinner Farias, and Bruno Silva. A survey on the use of uml in the brazilian industry. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering*, SBES '21, page 275–284, New York, NY, USA, 2021. Association for Computing Machinery.

[39] Alan C Kay. The early history of smalltalk. In *History of programming languages—II*, pages 511–598. Association for Computing Machinery, New York, NY, USA, 1996.

[40] Jonathan Lazar, Aaron Allen, Jason Kleinman, and Chris Malarkey. What frustrates screen reader users on the web: A study of 100 blind users. *International Journal of human-computer interaction*, 22(3):247–269, 2007.

[41] Claudia Loitsch and Gerhard Weber. Viable haptic uml for blind people. In Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler, editors, *Computers Helping People with Special Needs*, pages 509–516, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[42] Leandro Luque, Leônidas de Oliveira Brandao, Romero Tori, and Anarosa Alves Franco Brandao. On the inclusion of blind people in uml e-learning activities. *Revista Brasileira de Informática na Educação*, 23(2), 2015.

[43] Martin Mazanec and Ondrej Macek. On general-purpose textual modeling languages. In *Dateso*, volume 12, pages 1–12. Citeseer, 2012.

[44] Oussama Metatla, Nick Bryan-Kinns, and Tony Stockman. Auditory external representations: Exploring and evaluating the design and learnability of an auditory uml diagram. In *Proc. of the International Conference on Auditory Display. Montréal, Canada*, pages 411–418, 2007.

[45] David S Meyer and Steven A Boutcher. Signals and spillover: Brown v. board of education and other social movements. *Perspectives on Politics*, 5(1):81–93, 2007.

[46] Karin Müller. How to make unified modeling language diagrams accessible for blind students. In Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler, editors, *Computers Helping People with Special Needs*, pages 186–190, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[47] Juan Carlos Muñoz-Carpio, Michael Cowling, and James Birt. Framework to enhance teaching and learning in system analysis and unified modelling language. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 91–98. IEEE, 2018.

[48] Charles B Owen, Sarah Coburn, and Jordyn Castor. Teaching modern object-oriented programming to the blind: an instructor and student experience. In *2014 ASEE Annual Conference & Exposition*, pages 24–1167, 2014.

[49] Andrew J Oxenham. How we hear: The perception and neural coding of sound. *Annual review of psychology*, 69:27–50, 2018.

[50] Nicola Papazafiropulos, Luca Fanucci, Barbara Leporini, Susanna Pelagatti, and Roberto Roncella. Haptic models of arrays through 3d printing for computer science education. In Klaus Miesenberger, Christian Bühler, and Petr Penaz, editors, *Computers Helping People with Special Needs*, pages 491–498, Cham, 2016. Springer International Publishing.

[51] Vanessa Petrausch, Stephan Seifermann, and Karin Müller. Guidelines for accessible textual uml modeling notations. In *Computers Helping People with Special Needs: 15th International Conference, ICCHP 2016, Linz, Austria, July 13-15, 2016, Proceedings, Part I 15*, pages 67–74. Springer, 2016.

[52] Marian Petre. Uml in practice. In *2013 35th international conference on software engineering (icse)*, pages 722–731. IEEE, 2013.

[53] Marian Petre. "no shit" or "oh, shit!": responses to observations on the use of uml in professional practice. *Software & Systems Modeling*, 13:1225–1235, 2014.

[54] Rachel Potvin. The motivation for a monolithic codebase, 2015.

[55] Rebecca Reuter, Theresa Stark, Yvonne Sedelmaier, Dieter Landes, Jürgen Mottok, and Christian Wolff. Insights in students' problems during uml modeling. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 592–600, 2020.

[56] Michael Schutz and Jessica Gillard. On the generalization of tones: A detailed exploration of non-speech auditory perception stimuli. *Scientific Reports*, 10(1):9520, 2020.

[57] Aleksander Sek and Brian CJ Moore. Frequency discrimination as a function of frequency, measured in several ways. *The Journal of the Acoustical Society of America*, 97(4):2479–2486, 1995.

[58] Ather Sharif, Sanjana Shivani Chintalapati, Jacob O. Wobbrock, and Katharina Reinecke. Understanding screen-reader users' experiences with online data visualizations. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '21, New York, NY, USA, 2021. Association for Computing Machinery.

[59] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pages 364–371. Elsevier, 2003.

[60] Keng Siau and Poi-Peng Loo. Identifying difficulties in learning uml. *Information Systems Management*, 23(3):43–51, 2006.

[61] Andreas M. Stefik, Christopher Hundhausen, and Derrick Smith. On the design of an educational infrastructure for the blind and visually impaired in computer science. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, page 571–576, New York, NY, USA, 2011. Association for Computing Machinery.

[62] Harald Störrle. On the impact of layout quality to understanding uml diagrams. In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 135–142. IEEE, 2011.

[63] Biological Sciences Curriculum Study, National Institutes of Health, et al. Information about hearing, communication, and understanding. In *NIH Curriculum Supplement Series [Internet]*. National Institutes of Health (US), 2007.

[64] Clara Suied, Trevor R Agus, Simon J Thorpe, Nima Mesgarani, and Daniel Pressnitzer. Auditory gist: recognition of very short sounds from timbre cues. *The Journal of the Acoustical Society of America*, 135(3):1380–1391, 2014.

[65] Michael J Sullivan, Bruce Fairbairn, Charlie Shivers, LeAnna Parkey, WK Roberts, Sean Merril, and Matt Lea. Joint strike fighter: Restructuring added resources and reduced risk, but concurrency is still a major concern. Technical report, GOVERNMENT ACCOUNTABILITY OFFICE WASHINGTON DC, 2012.

[66] Danielle Albers Szafir, Rita Borgo, Min Chen, Darren J Edwards, Brian Fisher, and Lace Padilla. *Visualization Psychology*. Springer, 2023.

[67] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[68] Johan Wagemans, James H Elder, Michael Kubovy, Stephen E Palmer, Mary A Peterson, Manish Singh, and Rüdiger Von der Heydt. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172, 2012.

[69] Bruce N Walker. Consistency of magnitude estimations with conceptual data dimensions used for sonification. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 21(5):579–599, 2007.

[70] Bruce N. Walker and Gregory Kramer. Mappings and metaphors in auditory displays: An experimental assessment. *ACM Trans. Appl. Percept.*, 2(4):407–412, oct 2005.

[71] Hironori Washizaki, Masayoshi Akimoto, Atsushi Hasebe, Atsuto Kubo, and Yoshiaki Fukazawa. Tcd: A text-based uml class diagram notation and its model converters. In *Advances in Software Engineering: International Conference, ASEA 2010, Held as Part of the Future Generation Information Technology Conference, FGIT 2010, Jeju Island, Korea, December 13-15, 2010. Proceedings*, pages 296–302. Springer, 2010.

[72] Marcus Watson and Penelope Sanderson. Sonification supports eyes-free respiratory monitoring and task time-sharing. *Human factors*, 46(3):497–517, 2004.

[73] Ira Woodring and Charles Owen. An empirical study of user perception of audio stimuli in relation to a cartesian space. In *Proceedings of the 14th PErvasive Technologies Related to Assistive Environments Conference*, pages 8–15, 2021.

[74] Ira Woodring and Charles Owen. Results of preliminary studies on the perception of the relationships between objects presented in a cartesian space. *Technologies*, 10(1), 2022.

[75] Ira Woodring, Charles Owen, and Samia Islam. A method for presenting uml class diagrams with audio for blind and visually impaired students. In *Proceedings of the 17th International Conference on PErvasive Technologies Related to Assistive Environments*, pages 15–20, 2024.

[76] Stelios Xinogalos. Object-oriented design and programming: an investigation of novices' conceptions on objects and classes. *ACM Transactions on Computing Education (TOCE)*, 15(3):1–21, 2015.

[77] Jeong Yang, Youlg Lee, and Kai H Chang. Initial evaluation of jaguarcode: A web-based object-oriented programming environment with static and dynamic visualization. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 152–161. IEEE, 2017.

[78] Tsubasa Yoshida, Kris M Kitani, Hideki Koike, Serge Belongie, and Kevin Schlei. Edgesonic: image feature sonification for the visually impaired. In *Proceedings of the 2nd Augmented Human International Conference*, pages 1–4, 2011.

[79] Haixia Zhao, Catherine Plaisant, Ben Shneiderman, and Jonathan Lazar. Data sonification for users with visual impairment: A case study with georeferenced data. *ACM Trans. Comput.-Hum. Interact.*, 15(1), may 2008.

[80] Hong Zou and Jutta Treviranus. Chartmaster: A tool for interacting with stock market charts using a screen reader. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, pages 107–116, 2015.