UML Diagrams for Blind Programmers

Sarah Coburn
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48823
Email: coburnsa@msu.edu

Charles B. Owen
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48823
Email: cbowen@msu.edu

Many modern programming environments rely on conveying information to the programmer through heavily visual methods. One of these methods is the use of UML diagrams. These diagrams are intended to indicate relationships between classes or states in a software system. Objects are shown with lines connecting to other objects indicating specific types of relationships (such as object inheritance). By nature, this information is difficult to directly translate for blind programmers. While some programs exist to translate UML diagrams to a format a blind user can read, these programs often have flaws and cannot work with a wide variety of UML diagrams. In order to assist a blind student, we developed an auditory system to preserve the relational information from UML diagrams. We will also discuss essential information that blind programmers must receive from any UML translators, as well as future directions for programs that can translate a wide variety of UML diagrams.

General problem

UML diagrams are a commonly used resource in computer programming for representation of program structure. They can be used to represent classes and object relationships as well as state diagrams. Similar diagrams are used in other disciplines as well: general system block diagrams are used in a variety of engineering disciplines, as are circuit diagrams. These types of diagrams rely heavily on visual presentation. Essential information on relationships between elements is communicated through relative spatial position. Peripheral visual details help keep the user aware of general location of any single element compared to the whole system.

These diagrams represent a major hurdle for blind programmers, especially when moving beyond beginning programming skills. In early classes, navigation through simple program code is frequently accomplished through text to speech (TTS) readers. However, examination and creation of larger systems in later classes is frequently accompanied by the use of diagrams to accurately convey the system structure. In advanced classes and industry jobs, diagrams can also be used to bring new team programmers into an existing large-scale project. If this tool isn't accessible to blind programmers, it severely limits their ability to advance their skills and job marketability.

An eye tracking study[1] showed that when navigating UML diagrams, users fixate primarily on the nodes of UML diagrams, and have secondary fixations on the ends of connectors between nodes (which typically indicate type of relationship between objects). In other words, the primary focus is the major objects, followed by the relationships between them. Because these relationships can be complex and not purely sequential, textual interpretation of these diagrams can be difficult. Comprehension of the general flow of a diagram relies on visually contextual information.

One way to analyze a diagram is to treat it like a simple picture. Some research has been done to use Microsoft's InkAnalysis (normally used to interpret writing on a tablet device) to interpret drawn diagrams.[2] This approach allowed navigations among nodes in a diagram, but needed more work to associate text and types of relationships between objects. Another approach to communicating a diagram to a blind user is to have a tactile interface, so that the user can feel graphics.[3] However, this approach requires the extra tactile device. The TeDUB project attempted to convert arbitrary diagrams into a hierarchical spoken format.[4] However, the problem for UML diagrams is not recognizing that there are rectangles and lines, but recognizing that there are classes or states and connections among them and being able to rapidly follow the connections so as to comprehend the structure or sequencing.

Our approach

Prior to our current approach, we designed a tactile system to work with a blind student, intended to convey relevant spatial relationships between class or state elements. This is illustrated in Figure 1. These elements were represented using a combination of push-pins, rubber bands, and sticky notes that included class details in Braille (by using a portable embosser).

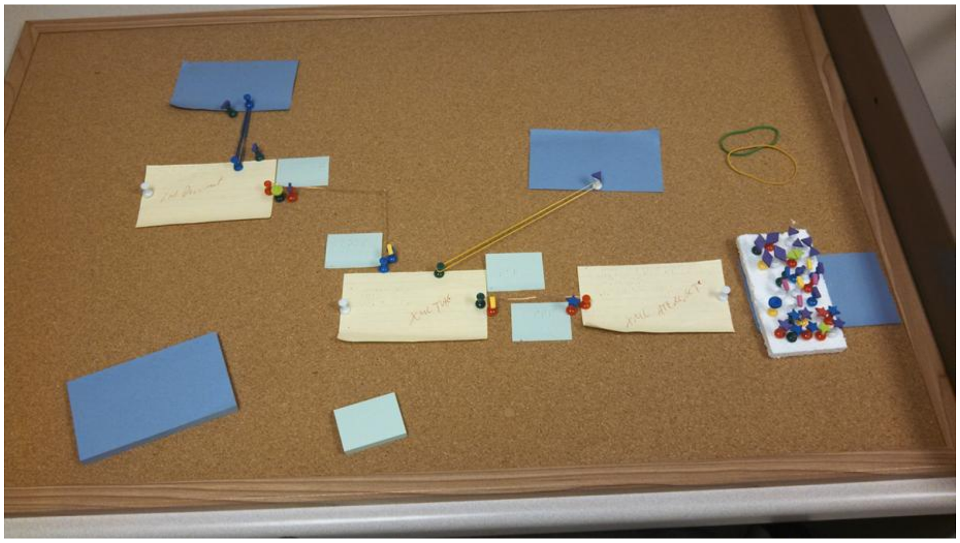

Figure 1 - UML diagrams represented for tactile interaction

Rubber bands were used to connect the elements so that the connection could be followed by touch. In order to designate specific relationships such as composition, nesting, and generalization (as are common in standard UML diagrams for programming), each push-pin had a shape that corresponded to each of these relationships, as shown in Figure 2. This allowed for easy composition of new UML diagrams.

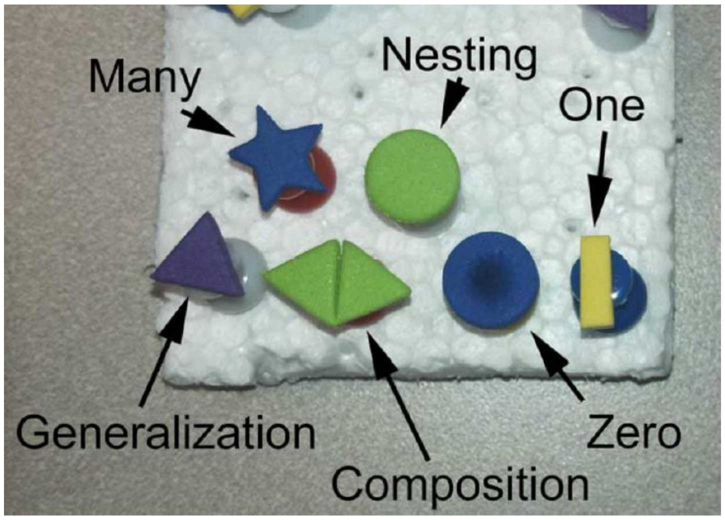

Figure 2 - Shapes representing class relationships

While this approach does have the ability to easily create new UML diagrams, it relies on an intermediate user to translate new diagrams. It also takes time to set up, especially with complex diagrams and creating new cards with Braille. This approach was useful for the short-term need of enabling a student to complete a course. However, in order to bring a blind user closer to the same efficiency of a sighted user, there should be no need for an intermediate user, and the user must be able to quickly interpret and create diagrams. As a result, we began a new project: Audible Browser.

Audible Browser reads XMI 2.1 files, specifically class and state diagrams (as produced by Visual Paradigm). The program interprets and presents the nodes of a diagram in a *tab group*. A tab group is a set of common items that can be read using the up and down arrow keys. The tab groups for a class are the name, attributes, operations, and associations. The name tab group initially speaks the class name and any inheritance or nesting; a summary of the class. The arrow keys allow the user to traverse through a list that includes any other class this class is nested in, classes nested in this class, classes this class inherits and classes that inherit this class. Hitting return on any of these lines allows the user to navigate to the associated class.

The next tab stop presents the class attributes. These are spoken as someone might read them aloud in a normal UML class diagram. For example, the attribute x: int=0 is spoken as “x colon int initially zero”. The left and right arrow keys allow the user to move from letter to letter, speaking the letters. This prevents ambiguity if the speech generation is not clear.

The next tab group allows the user to examine the operations of the class. The function func(x:int, y:int) : string is read as: “func parameters x colon int y colon int returns string”. As before, the left and right arrow keys read out the letters of the actual UML representation one at a time. The final tab group lists associations, aggregations, and compositions. In each case, the roles and multiplicities are read with the association. As an example: “association diagram 1 to state 0 dot dot asterisk”.

The physical structure of the diagrams is important. It infers relationships due to groupings and proximity.[5] Presenting nodes and providing navigation between nodes does not, in itself, convey the structure of a UML diagram as a 2D space with a physical layout that conveys information. To convey the structure of a diagram, Audible Browser presents a diagram using earcons, non-verbal sounds, in combination with speech. The current instantiation uses simple tones for each node in the diagram.

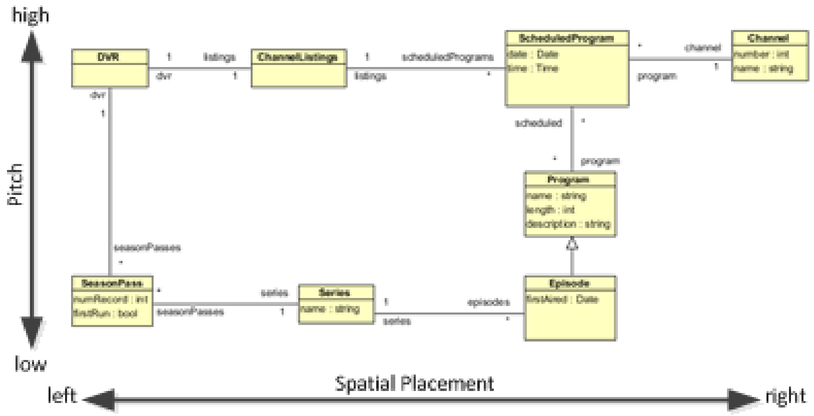

Figure 3 - Constellation presentation of 2D node placement

We refer to these tones as *stars*. A collection of stars, representing the nodes of the diagram, is a *constellation*. A UML diagram is conveyed to a user as a sequential presentation of stars. The left-right stereo placement of stars represents the X position in the diagram and the pitch, quantized to the nearest musical semitone, represents the Y position. Nodes that are higher on the page are presented with a higher pitch. The name of the node can also be optionally spoken when the constellation is presented. The constellation presentation is illustrated in Figure 3.

The use of stereo placement to convey horizontal placement is natural. Humans are able to resolve horizontal (azimuth) angular differences down to about one degree.[6] We are currently only implementing interaural intensity differences, as part of this research we will extend that to include interaural time differences, which model the differing arrival rate of sounds to the ear due to relative placement. Interaural time differences have been shown to be particularly important for localization of lower frequencies.[7]

While the ability of humans to place sounds horizontally is quite acute, the ability to place sounds vertically (elevation) is less so, even when the pinna (ear anatomy) is accurately modeled.[6] Instead, Audible Browser uses pitch to convey vertical placement of nodes. Mansur, Blattner, and Joy found that humans have a natural tendency to interpret higher frequencies as vertically higher in space.[8]

This presentation is on demand and continuous when navigating. Each star is presented for 400ms and the sequence is repeated. The presentation order is currently based on a sweeping pattern around the user, though this is a topic we want to explore in more detail. This provides a way to quickly convey the relative placement of the nodes around the user. The user navigates to other nodes using the mouse. All mouse movement is relative to when the movement begins, so there is no absolute placement and the mouse does not correspond to screen locations. Technically, the mouse is re-centered when it is released, so there is always space for movement in all directions and it cannot be clicked outside the window, accidentally inflicting a context change.

When moving, the left-right presentation of the stars changes. The pitch stays the same. A background tone is played as the constellation is continuously presented when moving. The tone provides a reference of the current Y location and varies as the user moves up and down. Humans are very poor at recognizing absolute pitches, so pitch is not a good representation of absolute position. However, humans are good at recognizing differences in pitch, so pitch is a powerful tool for representing relative placement.[9] Traversing to a node consists of moving so the sound is centered and the vertical motion tone matches the pitch. As nodes are neared, the node name is spoken. When the mouse button is released, Audible Browser snaps to the nearest node.

Future work

Our Audible Browser is a first example of how to present UML diagrams for browsing by a blind user. The two basic conventions in the system are the presentation of the content of a node and the presentation of the overall structure of the diagram using the constellation approach. The prototype for Audible Browser was created very quickly in order to provide a tool for a blind student during a course. Currently, it presents the constellation as a simple set of tones.

There are several enhancements that we propose for Audible Browser. Brewster demonstrated that earcons with more complex timbre and a musical base are more effective than simple tones.[10] We will replace the tones with a mix of instrument and synthetic sounds. These more complex sounds will help to differentiate diagram elements. Experiments will be conducted to select appropriate sounds and to refine the constellation presentation rate, which is currently fixed. These experiments will help to answer research question 2. As for the source code browser, a variable presentation rate can take advantage of human short-term pattern memory, allowing a faster presentation when browsing. Our student also indicated that speaking the names of the nodes intermittently during the constellation presentation helper her to better visualize the diagram structure.

The current Audible Browser solution presents the entire diagram in all cases. For larger diagrams, the audio clutter will become too great, and an efficient neighborhood selection approach will be necessary. Neighborhood selection heuristics will be devised. At first glance, it would seem that a simple range would be appropriate. However, diagram relationships should be taken into account, particularly groupings of content. Figure 4 illustrates a portion of a UML class diagram that implements the observer pattern.[11] A heuristic that includes DocumentObserver and ViewActors, but none of the other derived classes, would be confusing. A proposed neighborhood heuristic would cluster nodes into related groups and choose groups within a selected neighborhood distance.

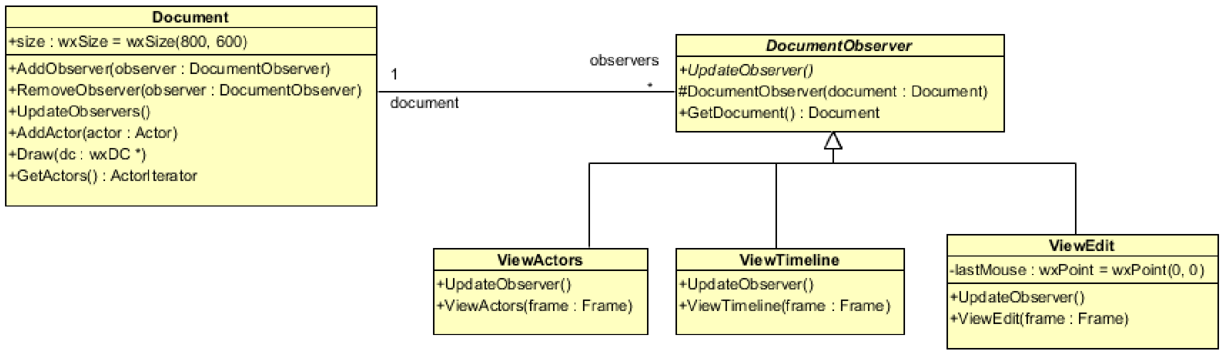
Document
+size : wxSize = wxSize(800, 600)
+AddObserver(observer : DocumentObserver)
+RemoveObserver(observer : DocumentObserver)
+UpdateObservers()
+AddActor(actor : Actor)
+Draw(dc : wxDC *)
+GetActors() : ActorIterator
1
document
observers
*
DocumentObserver
+UpdateObserver()
#DocumentObserver(document : Document)
+GetDocument() : Document
ViewActors
+UpdateObserver()
+ViewActors(frame : Frame)
ViewTimeline
+UpdateObserver()
+ViewTimeline(frame : Frame)
ViewEdit
-lastMouse : wxPoint = wxPoint(0, 0)
+UpdateObserver()
+ViewEdit(frame : Frame)

Figure 4 - Observer pattern UML example

Audible Browser allows users to browse quickly among connected nodes in a diagram. A single keystroke can transition from a class to associated classes, derived classes, or parent classes. The transition options are entirely dependent on existing links in the diagram. Eye-tracking studies have shown that users often shift visual attention to nodes that enjoy a *transitive relationship*, particularly though inheritance.[1] There is an association between Document and DocumentObserver in Figure 4, but there is also an inherited association between Document and ViewActors. A user who is attempting to comprehend how the actors are presented in the interface will likely be rapidly transitioning attention between these two classes. We will add the ability to make inherited and other transitive transitions to Audible Browser.

Adding the ability to edit UML diagrams will be an important addition. For the blind user, the associations and inheritance will not need to be as nicely formatted as in Figure 4. However, node placement is important. Studies have shown that the placement of items in a UML class diagram does impact program comprehension.[1] The ability to add nodes to a diagram will be relatively easy to add, but an efficient means of selecting multiple nodes and moving them around will be more complex.

Conclusion

Through Audible Browser we have established a method for presenting visual UML diagrams audibly to a blind user. Audible Browser automatically interprets the diagrams, circumventing the need for an intermediate user. While it is a primary approach, it provides a basis for future work and experimentation to allow blind users to cross another barrier to equal opportunities in computer science education and careers.

Bibliography

[1] Shehnaaz Yusuf, Huzefa Kagdi, and Jonathan I. Maletic, "Assessing the Comprehension of UML Class Diagrams via Eye Tracking," in *15th IEEE International Conference on Program Comprehension, 2007. ICPC '07*, Banff, Alberta, BC, 2007.

[2] Dennis Schwab, "Non-Visual Diagram Navigation Using Microsoft's InkAnalysis Tool," 2006.

[3] P. Parente and G. Bishop., "BATS: The Blind Audio Tactile Mapping System," in *ACMSE*, Savannah, GA, 2003.

[4] Martin Lorenz and Mirko Horstmann, "Semantic Access to Graphical Web Ressources for Blind Users," in *3rd International Semantic Web Conference (ISWC2004)*, 2004.

[5] D. Sun, "On evaluating the layout of UML class diagrams for program comprehension," in *Proceedings of the 13th International Workshop on Program Comprehension, 2005. IWPC 2005.* , 2005.

[6] Jens Blauert, *Spatial Hearing - Revised Edition: The Psychophysics of Human Sound Localization*.: The MIT Press, 1996.

[7] Frederic L. Wightman and Doris J. Kistler, "The dominant role of low-frequency interaural time differences in sound localization," *Journal of the Acoustical Society of America*, vol. 91, 1992.

[8] D. L. Mansur, M. Blattner, and K. Joy, "Sound-Graphs: A numerical data analysis," *Journal of Medical Systems*, vol. 9, pp. 163-174, 1985.

[9] Stephen Brewster, "Nonspeech auditory output," in *The human-computer interaction handbook*. Hillsdale, NJ, USA : L. Erlbaum Associates Inc., 2002, pp. 220-239.

[10] S. A. Brewster, P. C. Wright, and A. D. N. Edwards, "A detailed investigation into the effectiveness of earcons," in *Proceedings of ICAD'92*, Santa Fe, NM, 1994.

[11] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*.: Addison-Wesley Professional, 1994.