A Sketch/Speech Interface and Usability Study for Software to Teach and Learn Concurrent Programming

by

Daniel Schlegel

Submitted in partial fulfillment of the requirements for the degree of
Master of Arts
in
Human Computer Interaction
at the
State University of New York, College at Oswego
July, 2009

Table of Contents

Background

The Tabcon Software

Tabcon is designed as a teaching and learning tool for concurrent programming. Originally conceived by Dr. Lin Qiu the project uses sketch input from a Tablet PC to aid in the creation and simulation of diagrams representing deadlock situations, a common issue in concurrency (Qiu, Tabcon: Tablet-based Tools for Teaching Concurrent Programming, 2007). It is meant for use both by students and by professors. Presentations traditionally by professors on this particular aspect of concurrent programming result in messy chalkboard diagrams and later incomprehensible student notes. Using this software the student can take notes and save their notes and simulations for later review, while the professor can show the simulation to the class as an instructional tool – even saving them and placing them online for students to download.

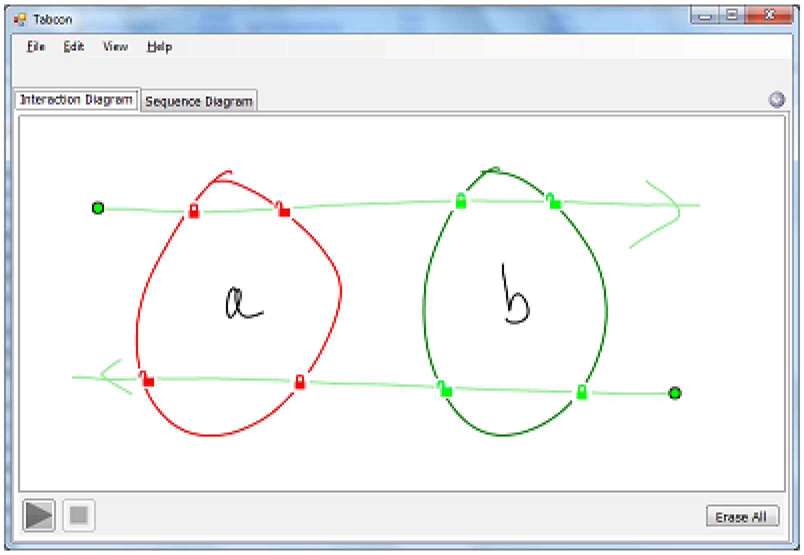


Figure 1: The Tabcon Interaction Diagram

Tabcon is meant to provide an interface that is very similar to that of a whiteboard – with very few interface elements. The main feature of the software – the interaction diagram – provides a blank white canvas on which to sketch with controls to manipulate the simulation at the bottom. The user of the software draws circles to represent objects or sections of code and drags the properly colored locks to the threads which pass through the code segments to signify critical sections. This is a fairly universal metaphor through which the deadlock problem can be explained. The user can also annotate each of the locks to signify the order in which they must occur and run simulations. Other features of the software allow locks to be annotated with code as well. When a circle is locked it is shown as bold and the green ball which shows simulation progress on the non-lock holding threads will not enter the circle. The simulation will alert the user if it cannot complete successfully due to a deadlock problem.

After observing professors attempting to use the application in a presentation setting and after making attempts to do so myself it became obvious that while the sketch recognition provided on the Tablet PC works well in a controlled setting, it does not perform as well used in front of a classroom. In a controlled setting such as using the software at a desk where complete concentration is directed towards the sketch the accuracy of the shapes and gestures drawn appears to be much higher than in the conditions of a teaching environment. While the professor is talking they shouldn't need to be concerned with whether their sketch was recognized properly (that is, whether the software recognized the gestures drawn as they were intended). In order to attempt to ease the pain so to speak it was hypothesized that Tabcon could benefit from multimodal speech/sketch input interface. The interface would recognize the speech of the presenter and apply it to the recognition of the sketch elements when the in-built algorithms failed.

Sketch

The idea of interfacing with an electronic device via a pen interface has existed for over a hundred years (Gray, 1888), but only in the past 15 years has it become mainstream. First entering mainstream culture with the Apple Newton and Palm Pilot in the early 90s interfaces which used the pen for input were largely reserved for mobile PDA devices. The first consumer pen tablet computer was introduced in 1992 by IBM (Smith, 2007). Up until the turn of the century though these devices were very rudimentary in their pen input – they allowed the pen to be used like a mouse would be for basic point and click type actions, and they incorporated early handwriting recognition and gesture technologies.

Sketchpad is commonly regarded as the first system which both provided a graphical user interface and allowed the user to draw on the screen. A light-pen is used to position objects on the screen and to manipulate them (Southerland, 1963). Among the earliest examples of a system to allow input to a computer via gestures on the screen is that of Fingerpaint. This allowed manipulation of objects on the screen using basic gesture recognition of the input (Minsky, 1984). This research was enhanced much further around 1991 when interest in this area accelerated, most likely because of commercial interest in handheld electronics. Research in single stroke recognition of sketches on the screen was developed around this time (Rubine, 1991). Multiple stroke recognition of geometric shapes (Apte, Vo, & Kimura, 1993) was the next major stride and made future naturalistic sketch input systems possible.

Naturalistic type sketch systems allow a mostly unconstrained input system within a domain. The input types allowed are obvious because of the domain the software package falls into (Schlegel & Tenbergen). Two examples of systems which share these characteristics (besides Tabcon) are SketchUML and ASIST.

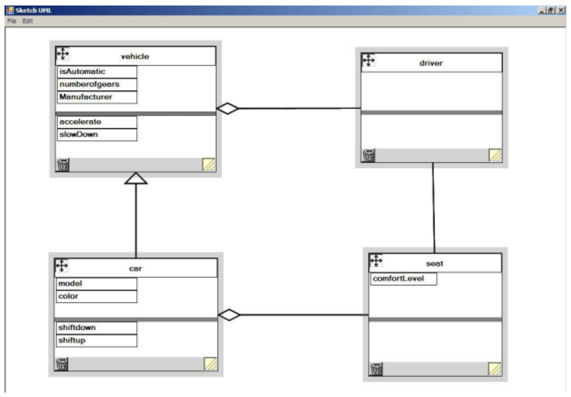


Figure 2: SketchUML's minimalistic interface

SketchUML, much like Tabcon, is a Tablet PC application which is designed with a very minimal user interface. It allows the student or professor to draw a UML diagram as they would on paper on the screen of the Tablet PC. Hand drawn squares, handwriting, and connective lines are recognized are converted into container boxes, text, and the appropriate type of connection. The user can then rearrange the diagram, make corrections, or even submit it for critique by a teacher. (Qiu, SketchUML: The Design of a Sketch-based Tool for UML Class Diagrams, 2007)

ASIST is a research project developed for early sketch design of physics related systems. The user can draw on the screen representations of physical objects and have them behave in a realistic way. For example a circle representing a wheel can be drawn at the top of a wedge representing a ramp. When the simulation is begun the wheel would roll down the ramp as it would in the real world. (Alvarado).

These interfaces aren't restricted to serious applications - games are beginning to use a naturalistic sketch interface as well. One of these interfaces similar in nature to ASIST is Crayon Physics (Kloonigames). In Crayon Physics the user tries to solve puzzles by drawing objects which exhibit physical properties (for example objects fall from the sky when drawn off the ground, wedges work as ramps, etc). After drawing these elements, the player draws a mechanism to start a ball running through the created elements to reach a goal.

In order to create applications which use pen input generally some sort of software framework is used since the amount of research and work to create a system from scratch is a severe barrier to entry. In the past there have been a few projects started to provide frameworks for designers and programmers alike. The classic method for sketch recognition systems is for them to use a recognizer based on that developed by Dean Rubine – where the gestures are learned through example (Rubine D. , 1991). Interfaces were created to let designers easily do this - Quill lets a designer create gestures for use in a sketch based application through a graphical user interface. The designer can draw the gesture required for their application and this serves as a prototype against which sketches in the application are compared against (Hong, Landay, Long, & Mankoff, 2002). Other systems such as LADDER (Hammond & Davis, 2005) and its derivatives (Corey & Hammond, 2008) allow designers to define how their sketches are drawn, displayed, and edited via a user interface. These definitions can then be translated into a set of recognizers to be used in software.

Most commonly used in more commercial type settings and to some extent in research is Microsoft's Tablet PC API which is based on the .NET framework. Since this is integrated with a current language system already no other tools or specific skill sets outside of those a programmer might already have are required to create sketch systems which utilize their built in set of gestures and handwriting analysis tools. Since this system is closed source though, it is hard to determine which types of recognition systems are employed (Microsoft Corporation).

Speech

Speech recognition technology falls generally across two primary uses. The first of which is what first made the idea of speech recognition attractive and that is dictation. The idea that someone could speak to a computer and have it recognize speech and transcribe it in to text for you was one which led many companies to invest in speech recognition. One field in which this has been and still is quite common is that of medical transcription. The second and currently more common use of speech technologies is command recognition – similar in nature to when you speak on the phone to an automated menu system. In these systems there are generally only a few options available so the issue of differentiating between which of them the speaker is saying is a less complex operation. This type of recognition was the earlier of the two developed – in the 1950s Bell Labs developed a speech recognizer which was able to discern numbers (History of Voice Recognition Technology, The, 2004).

In the 1970s ARPA studied the speech recognition problem further and found that the goals of the projects needed to be shifted. Up until this point the focus was on discrete speech recognition – that is the recognition of one word at a time – but what was really needed was a continuous speech recognition model.

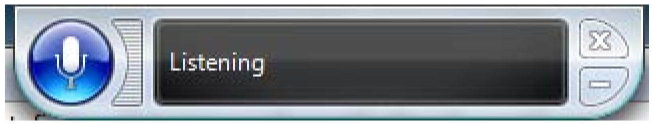


Figure 3: Speech Recognition in Windows

This type of model takes into account more than just one word in processing (History of Voice Recognition Technology, The, 2004). The first company which managed to release a commercial product to facilitate continuous speech recognition for dictation was Covox in 1982 (Dragon Systems). Most of the systems in use today use some system to sample the input at a high rate to determine the phonemes in use in words or phrases in combination with a Hidden Markov Model (HMM). These models use the probabilities of one phoneme following another to help disambiguate the input and a set of legal accepting states. It is assumed though in any Markov Process that the probability of any next state relies only on the present state and no past states (known as the Markov Property). This implies that in a HMM the probability of the next phoneme is only dependant on the current phoneme. There are parameters between states in the HMM which define this probability, and while these are known the sequence of states used to reach a terminal state is not in a Hidden Markov Model. The word phrases recognized (and their possible alternatives in the ambiguous cases) can be compared against n-gram models which can help predict which word sequences are more likely given a small amount of context (Jurafsky & Martin, 2000).

These commercial applications all use a proprietary recognition system which were developed in-house or licensed. For someone who wishes to create software which uses speech recognition there are not an excessive amount of options. The only one of these APIs which is truly portable between computer operating systems and architectures is the Java Speech API (Sun Microsytems). Platform specific APIs also exist such as the Microsoft Speech API which provides many of the same functions as the Java Speech API. There are other platform specific APIs which are less general though – for example the Apple Speech API is useful only for command recognition (Apple Computer).

Multimodal Interfaces

Multimodal interfaces can refer to two distinct sets of interfaces – those with multiple output modalities and those with multiple input modalities. Multiple output modalities are often used for software packages which cater to the disabled in some way but they are also used in our everyday computing. For example in Internet Explorer going to a website invokes a “click” noise coming from the speakers to cue to the user that they have reached a new website. Other possible modalities besides those of vision and sound can include tactile (touch) feedback, or even olfactory feedback (Sarter, 2006).

Interfaces which use more than one input modality have existed for quite a few years and are of our primary concern here. A common example is the use of speech in addition to the traditional keyboard and mouse inputs. Interfaces such as these are very common and exist not only on our desktop computers but oftentimes in our portable devices as well. In the past ten years there has been an effort towards sketch recognition as well. Both speech and sketch are extremely natural input methods if they are designed properly – that is, if they are designed in a natural way not forcing the user to learn some arbitrary command set.

Although voice controlled telephone menu systems are still a source of frustration, speech recognition technology has come a very long way in the past few years. Most recognition engines require a brief training period to understand your voice but after that are accurate to a very high percentage. Some applications such as Dragon Naturally Speaking Preferred Edition claim to have recognition results as high as 99% and not require any training to get started (Nuance), although these systems average around 90% efficiency.

Multimodal interfaces which utilize both sketch and speech are rare, and those which aim to do so in a natural way are even more so. One of the first interfaces to attempt to combine the two modalities is that of QuickSet. This program uses both pen-based gesture input along with the use of specific speech commands to perform an action. The goal was to decrease errors in recognition from a speech-only interface in both a stationary and mobile setting. The study found that speech was the less reliable of the two interface technologies but that the combination of the two produced results better than either one individually (Oviatt, 2000).

The only example of a naturalistic multimodal speech/sketch interface which exists currently is that of the ASIST application created by Adler and Davis at MIT's CSAIL. As previously mentioned, ASIST is a software system which recognizes mechanical properties of drawn objects and allows them to interact physically. The addition of speech recognition and processing to this involves the application performing certain actions when pre-programmed phrases are mentioned – allowing a modification of the diagram based on spoken input. The example provided allowed the user to draw pendulums of a Newton's Cradle and arrange them properly solely through the use of speech. In this example the pendulums drawn were constrained in size and location through speech alone, as shown in Figure 4 (Adler & Davis, 2004).

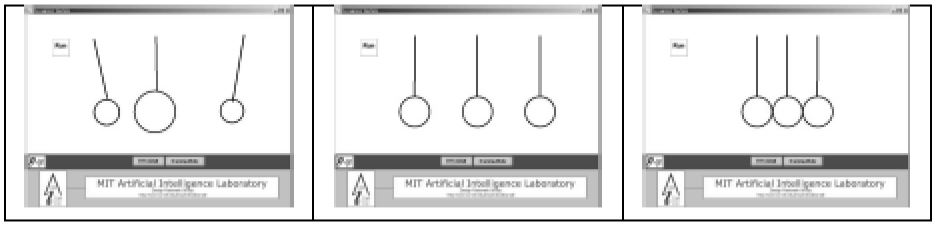

MIT Artificial Intelligence Laboratory
MIT Artificial Intelligence Laboratory
MIT Artificial Intelligence Laboratory

Figure 4: The changing model based on speech recognition in ASIST

TabconV2

Goals

The goal of TabconV2 in this specific context was to provide a system which could be usable for teaching basic issues in concurrency. Whereas the initial version of Tabcon focused both on the lecturer and the student, this iteration was largely interested in the teaching aspect. In teaching there are many issues – oftentimes the professor is not sketching to the best of their ability since they are multitasking – speaking, thinking, and drawing. They are also not always drawing on a medium as familiar as that of paper. The goal in this project was to use a multimodal speech-sketch interface to alleviate the issues which exist in using a tablet type computer system as a teaching tool in front of the classroom.

Since the sketch itself was already something that worked as a method to convey the information to the student and the issues with the system were in the area of recognition it was decided that speech should be used to augment the recognition capabilities. This is in contrast to the ASIST system created at MIT which used specific phrases to manipulate objects in the sketch; the primary concern of TabconV2 was to use speech as a method of recognition – not manipulation. Therefore the goal was to use specific words and phrases which either refer to a class of objects in general or are mapped to a specific reference of an object to assist in the recognition of objects drawn on the screen.

Architecture

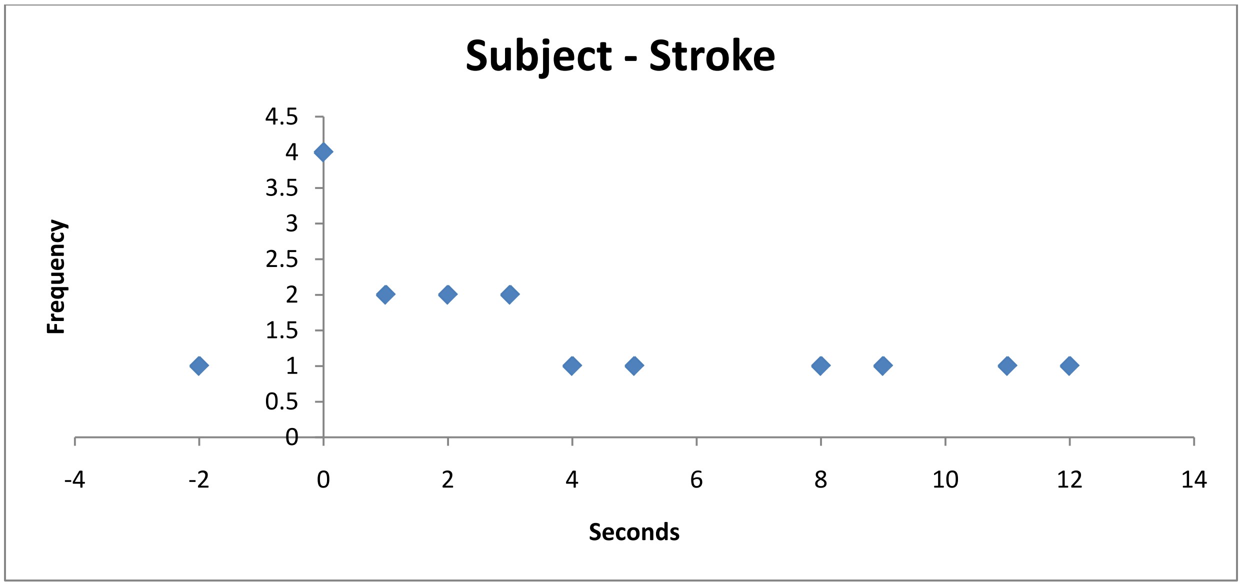
In developing the architecture of the speech input portion of the system, it was decided to use the Microsoft System.Speech toolkit which is part of .NET 3.0 and onwards. This framework abstracts the default speech recognition system being used on that PC to a single set of APIs. This allows the user to change out what recognition toolkit they are using to one which works best for them. On older Windows XP machines with no additional software installed this defaults to the Microsoft Speech API 5.1 and on later versions of Windows it defaults to version 5.3. Therefore the speech recognition is guaranteed to work on any Windows computer made in the last 8 years.

The overall architectural plan which was developed for TabconV2 is pretty straightforward. As the presenter speaks their words are cataloged. As the presenter is sketching, when an issue comes up where the sketch element was not recognized the speech data is referred to in an attempt to gleam some extra information about what the lecturer was sketching. In practice this is not quite so simple. Speech input frameworks recognize their input in phrases, so it is necessary to access to some number of the previous and future phrases to help analyze the sketch. Of course the system shouldn't take into consideration old data, or still process once a result should have been perceived. Therefore it was necessary to determine how much time before or after the sketch element has been drawn that speech need be considered.

Adler and Davis have studied this particular aspect of multimodal input briefly. They found that in most cases sketching precedes the mention of the subject of the drawing, yet the sketch generally is started after the phrase containing the subject has begun. This study was done with an experimenter and participant sitting at a table with their Tablet PCs (Adler & Davis, 2007). In order to adapt this data to the needed applications in TabconV2 I decided to determine whether this was consistent with teaching a lesson on deadlocks and basic threading to a class on a Tablet PC in front of a classroom. This is different than sitting at a table since the presenter is generally standing and is attempting to present their work to a room instead of the more intimate discussion style employed by Adler and Davis. The methodology of Adler and Davis is also different than what might be traditionally experienced with a professor writing on a whiteboard – which forces the professor to face away from the classroom while they write or draw. Writing on a tablet is more akin to an overhead projector – the lecturer faces the classroom while writing or drawing on some surface which is displayed to the room.

In order to determine both the order in which speech and sketch occurred and the amount of time in which they differed several videos of lectures on deadlocks taught from a Tablet PC were used from the archive of lectures available at both iTunes U and YouTube. The different professors analyzed used different teaching techniques and metaphors to describe the process, but the lecture was nearly always accompanied by some sort of sketch.

Much as Adler and Davis did, the time the phrase began, the time the subject of the drawing was mentioned, and the time the sketching began were all cataloged. Phrase here is defined the same way the speech recognition systems would define it – it is either the beginning of a sentence or the closest pause in verbalization.



The chart above shows the number of seconds after the stroke was drawn that the subject of the stroke was mentioned verbally. These results are in agreement with those Adler and Davis have shown, but here we can predict the amount of time after drawing that the user might speak using a Tablet PC in a lecturing situation. The average time was 3.47 seconds, and 75% of all times were less than five seconds. The times which were 8 seconds and greater were when the speaker was preparing the drawing for the next part of the lecture but was still talking about a previous topic. Based on these findings the time frame of +/- 5 seconds was chosen to analyze speech before and after the stroke was drawn.

Qualitative Issues

The next step in order to process the speech was to determine how exactly the lecturers referred to the diagrams in question. The objects this project was most concerned with were the threads (lines with arrows in the TabconV2 software) and objects (circles) since these are the two principle elements of these diagrams. The same videos used earlier were again analyzed to determine exactly that. The professors observed in these videos have speech patterns related to the diagrams which seem to fall into two distinct groups.

The first group drew the shapes which comprise the diagram but referred to them by the names of the abstract objects they represented. For example one professor on drawing a circle to represent an object referred to it as a hashmap. As the lectures progressed this group appeared to always refer to that object as its abstracted name and never by its shape. When drawing a second object or thread, they tended to say “another one”, “another hashmap”, or “you now have two threads.”

The second group of lecturers tended to refer to the sketch objects by their shape and a less abstracted name, for example “this is an s protected box” where “s” would refer to a lock and the box would refer to an object. This style seems to be intrinsically more difficult to understand by a human because of the lack of relationship to familiar objects, but for processing this case is actually easier.

Combining These Results

From the information about the periods of time which are valid for processing speech and sketch data and the information about what the speech patterns look like in relation to these diagrams a more detailed architecture was arranged.

For the given example professor who refered to his sketch objects by their reference names a special table of references needed to be created and maintained. It was decided that it should contain the reference names of objects and some way to refer to the actual stroke which was drawn. For example if a circle representing an object is drawn and the name hashmap is written inside it, the name hashmap needs to be mappable back to that specific circle. This solves direct reference problems and simplifies recognition of an object when the “you now have two hashmaps” scenario occurs.

A table which acts as a list of actions needed to be maintained as well. This allows the lecturer to draw two or three objects referring to each of them as “another one.” Given this kind of data if recognition fails on one of the objects it is obvious to the analysis engine which type of object to treat it as.

Of course this process of building these tables and lists, then checking them against failed sketch recognition was designed to happen in the background so as not to affect the user. An issue that may occur here in specific cases would be that the delay of the sketch recognizer combined with the delay of speech recognition and analysis of speech that happens after the sketch is drawn is so large that the user might think something is wrong. This issue may require further analysis.

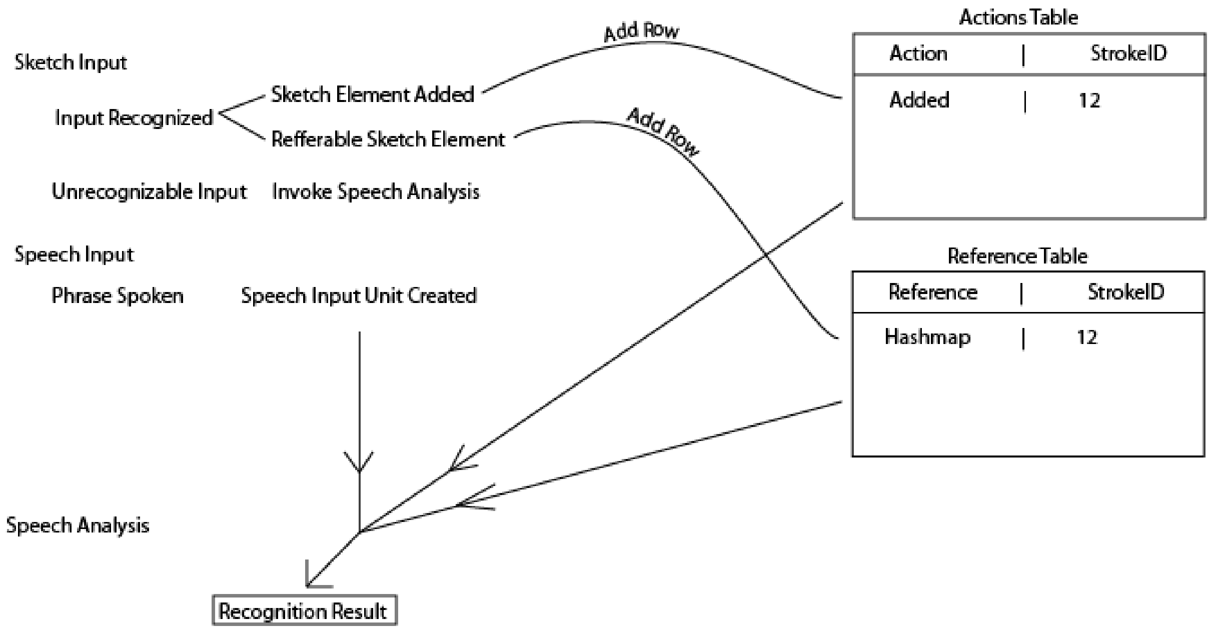


Figure 5: A view of the processing flow

Completed Form

Tabcon was designed to fit into a set of constraints defined by Schlegel and Tenbergen in an in progress work that are meant to define what they call a "Naturalistic and Minimalistic Interface." Interfaces such as these apply the theories of logical, physical, cultural, and semantic constraints (Norman, 1988) to computer software (Schlegel & Tenbergen). In other words, knowing basic knowledge about the constructs is all that is needed to be able to use a software system such as Tabcon. The addition of speech recognition only enhances the compliance of this project to these specifications. In theory, no longer are there extremely rigid constraints on what the diagram has to look like if it is accompanied by speech which explains the diagram in an expressive way.

Because of this interface constraint there is not very much to speak of the user interface of Tabcon – and this project has added literally nothing to it. Everything that happens in respect to the speech analysis and the coalescence of the two technologies is completely transparent to the user – as it should be. The lecturer using the software should be able to focus on their lecture without any interruptions and that is what this project provides.

Preliminary analysis of the multimodal interface (based data obtained during task 3 results of the usability study in the next section) found that the contribution of the speech interface in the overall usability of the TabconV2 software was not insignificant. Diagrams were recognized based on sketch alone 83% of the time, with speech recognition assisting 41% of the unrecognized strokes. This brings the recognition percentage to nearly 90%.

Usability Study

In order to determine the usability of the TabconV2 application, a usability study was performed.

Executive Summary

The purpose of this usability study was to analyze the overall ease of use of the TabconV2 software both with and without training using the software. Three key aspects were tested – the ability of the user to use the software with no prior knowledge, to create basic diagrams, and to create advanced diagrams with instruction. The usability test was run in the HCI Lab at SUNY Oswego with eight participants. Each participant was asked to complete the three tasks along with accompanying surveys. From the results collected it was found that the overall impressions of the software were positive with most participants finding it a natural alternative to using a whiteboard with only minor recognition issues and feature requests. Each participant had their own teaching style and used the software in their own individual way. In general the participants reported the software was largely easy to use and that they found it to be a natural interface metaphor.

Methods

The user tests were completed by Daniel Schlegel under the supervision of Dr. Bernadette Sibuma. The usability study was approved by the SUNY Oswego Human Subjects Committee and all participants were given informed consent. The study was conducted July 9th and 10th, 2009.

Participants

In order to qualify for the usability study, the students involved were required to be majors of Computer Science, Information Science, or one of the associated majors. A convenience sample was used in choosing participants. For all demographic information regarding the participants please refer to Table 1. Please note that in ranking Tablet PC experience a 1-5 scale was used.

Participant ID	Age	Gender	Major	Year	Tablet PC Experience	Experience in Concurrency?	Knowledge of Deadlocks?	Concurrency course?
1	23	M	Comp. Sci.	Sr	5	Y	Y	N
2	19	M	Comp. Sci.	So	1	N	Y	N
3	19	M	Comp. Sci.	So	2	N	N	N
4	20	M	Comp. Sci.	Jr	3	Y	N	N
5	19	M	Comp. Sci.	Jr	3	N	N	N
6	21	M	Comp. Sci.	Sr	4	Y	Y	Y
7	20	M	Comp. Sci.	Sr	3	N	N	N
8	20	F	Cog. Sci.	Jr	1	Y	N	N
Average	**20.125**				**2.75**			

Table 1: Participant Information

Measures, Testing Environments, and Procedures

Each task was carried out in the HCI Lab at SUNY Oswego. The tasks each had slightly different measures and procedures based on what they were attempting to test.

Task 1

Prior to the first task in order to ensure that participants were on equal footing with regard to the concepts which the TabconV2 application is used to illustrate in teaching a class the participants were each shown a short eight minute video explaining the problem using a traditional whiteboard type technique before completing the first task. They were allowed to repeat watching the video as many times as needed until they felt they were comfortable with the material.

The first task aimed to test whether a user with no specific knowledge of the TabconV2 software could create a very simple diagram with no prior instruction on the use of the program or its specific conventions. The participants were introduced to the concept of a Tablet PC and the functions of its pen if they had no prior knowledge.

After completing this initial training the participants were given the following task:

> "Create a diagram which contains an object and two threads which pass through it. As you create the diagram, explain it to the classroom. "

This task was analyzed based mostly on the computer controlled result recording and the diagram produced.

Task 2

Prior to the second task the participant was shown a brief video of a similar lecture to the one shown previously, only this time the lecture was conducted using the TabconV2 software. This introduces the participant to the elements of the software which they may not have been able to ascertain without any instruction and introduces them to further features such as simulations and the method for using locks.

After completing the instructional video the participants were asked to perform the following task:

> "Create a diagram which contains two objects, two threads, and four locks in a way which **will not** produce a deadlock situation. As you create the diagram explain what you are doing to the classroom. When you have completed the diagram run the simulation."

Task 3

The final task did not require any instructional video as the one previously already covered these concepts. The participant was asked to complete the following task:

> "Create a diagram which contains two objects, two threads, and four locks in a way which **will** produce a deadlock situation. Be sure to annotate the locks with numeric order values. As you create the diagram explain what you are doing to the classroom. When you have completed the diagram run the simulation."

Results

Since the actual carrying out of the tasks themselves were not being tested, but more the ability of the software to carry out the tasks with each individual users style, not many quantitative metrics were used. The only metric which was logged manually for the tasks was the number of times the user had difficulty in each task and how they corrected these problems. How the user corrected their mistakes was only logged for tasks two and three since for the first task the user was not expected to be aware of how the software functioned and how to correct errors.

The following table contains data for the first task only:

	Circle Recognition	**Thread Recognition**	**Order Recognition**	**Other Difficulty**
Avg. No. Difficulties	0.125	0.75	■	0.125

The following table contains data for the second task only:

	Circle Recognition	**Thread Recognition**	**Order Recognition**	**Other Difficulty**
Avg. No. Difficulties	0	0.5	■	0.0625

The following table contains data for the third task only:

	Circle Recognition	**Thread Recognition**	**Order Recognition**	**Other Difficulty**
Avg. No. Difficulties	0	0	1.1	0

The following chart displays information collected in the post task surveys regarding the participants' overall experience with each task. All participants were able to complete the three tasks with varying levels of efficiency. On average, participants felt that the interface did not interfere with the tasks and that the software was natural and met expectations. All participants felt the video demonstration using the software shown before task two was helpful.

	Task 1	**Task 2**	**Task 3**
Completed Task	1	1	1
Interface Obtrusive	0.125	0.25	0.25
Met Expectations	0.875	1	0.75
Easy to Use	0.875	■	■
Is it Natural	■	0.875	0.875
Did the Video Help	■	1	■

Preliminary Analysis

Task 1

This task was the least involved of the three, but tested the "out of the box" usability of the application the most.

Issue: The most common mistake a user made here was that the arrows they drew on the thread lines either did not cross the initial thread line (as the program requires for recognition) or they drew arrows as one stroke. The following three figures show an acceptable arrow head and two additional thread diagrams which did not work. Occasionally the speech recognition helped in these areas, but not 100% of the time.

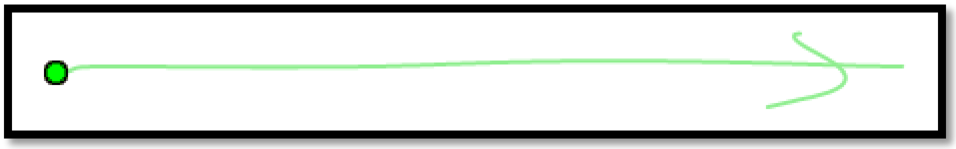


Figure 6: A "correctly" drawn thread

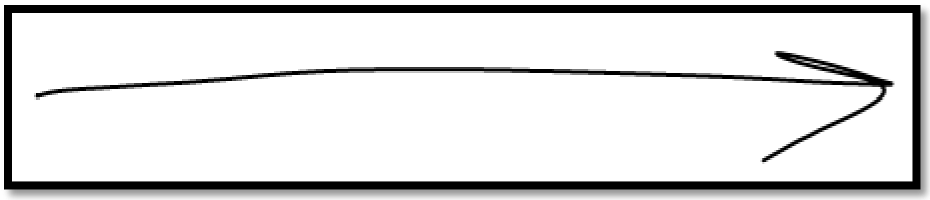


Figure 7: This thread did not work because it was only one stroke.

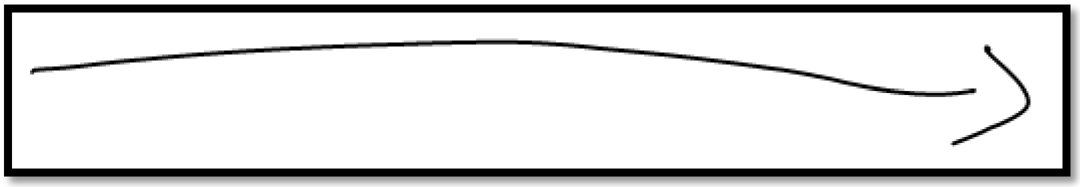


Figure 8: This thread did not work since the arrow did not cross the thread.

Task 2

Participants had the least difficulty with this task.

Issue: Users occasionally missed the lock image when attempting to drag it. The lock and unlock images do not appear to be large enough to be accurately dragged with the pen in a lecture type situation. This resulted in excess lines on the drawing which the user had to then go back and erase. Following this the users affected were more careful in dragging the locks to the threads.

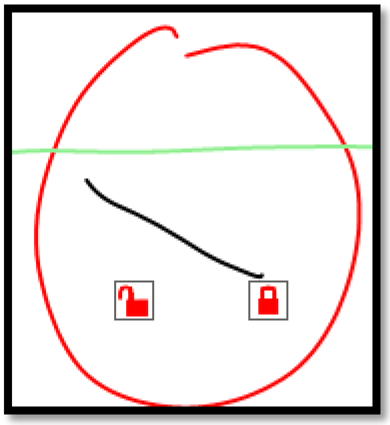
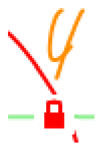
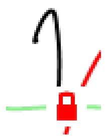
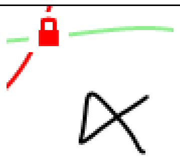


Figure 9: This user has just missed the lock when attempting to drag it to the thread.

Task 3

The third task was the most difficult and frustrating for the users as was expected – it uses nearly all of the features of the interaction diagram.

Issue 3: Numeric values for the order of execution of threads were not always recognized. Some users had no problem at all and others had many issues. The problems encountered were users who drew their numbers on a severe slant or a European style number one.

 Figure 10: A recognized number four	 Figure 11: An unrecognized European number one	 Figure 12: An unrecognized 4 on a severe slant

Major Findings

Handwriting Recognition

Some users had issues with the handwriting recognition in relation to the writing of execution order numbers. Some users experienced no issues at all while others had major issues. The users who tended to write their numbers in a conventional or "correct" manner did not experience any problems. Participants who wrote their numbers in ways which were non-standard (even in relation to stroke direction or style) often had difficulty and had to rewrite their numbers numerous times.

Gesture Recognition

Users when placed in a situation where they did not know how to use the software oftentimes drew the thread diagrams in a way which was not recognized. This had to do with the method by which the arrow of the thread strokes were drawn. Since users in this situation did not know how to use the application they did not correct the mistake. In subsequent tasks in which they had seen how to draw the arrows, there were no issues.

Annotations Cause Problems

Some participants attempted to annotate the drawing both with text and with additions to the diagram which are not part of the intended use of the diagram. These annotations were occasionally recognized as diagram objects, occasionally as nothing, and sometimes as execution order annotations.

Simulation Control

Two participants mentioned that until they saw them used they were unsure what the simulation controls might be used for. The play/pause/stop controls appear to only be a good metaphor when there is a label of some sort to demonstrate what they do.

Small Draggable Objects

Objects which were draggable were oftentimes too small for the user to reliably target with the pen tip while using the application in front of the class. When this mistake was made users tended to end up with a line they did not mean to draw in the sketching area. All users erased this line and were more careful in selecting the small objects in trying to drag the objects again.

Recommendations

Issue 1

Drawing diagrams and writing text both yield poor results in some circumstances.

In task 1 users were asked to draw a diagram without previous knowledge of the application. In task 3 users were asked to create a diagram which produced a deadlock.

Problem 1A

As shown in figures 11 and 12, some users consistently had issues writing numbers which were recognizable by the software for use in the execution orders.

Recommendations

In order to remedy this issue I recommend extending the speech recognition system to assist with the recognition of order of execution strokes. The data collected from the speech transcripts tends to suggest that most users mention these orders in their verbalization of the process.

[User reports: When a participant was asked the question "What frustrations did you have today with the software?" they wrote "It wouldn't recognize my numbers"]

Problem 1B

Users had to modify how they drew their arrows from the first task to the second task after seeing what kinds of arrows the software would accept in drawing threads. This goes against the natural user interface goals of this project.

Recommendations

The acceptable drawings for a thread should be expanded to include a single-stroke line and arrow combination (as seen in Figure 7) and an arrow head which does not cross the line portion of the thread stroke (see Figure 8).

[User reports: When a participant was asked the question “Do you feel the software acted as expected?”, the participant selected no and wrote “I had to change my arrows”]

<u>Issue 2</u>

The software does not provide the flexibility to annotate the diagrams

Both tasks two and three provided real world examples of uses of the software.

Problem 1

It is currently impossible to annotate the diagram with symbols, text and markings which are not meant to be considered part of the simulation diagram.

Recommendations

Since it is used nowhere else in the software, pressing the button on the side of the pen could allow the user to enter an annotation mode. There should be some indication on the screen that this is the case as well – whether it be a message written somewhere on the screen or changing the cursor used to draw with from a dot to a pen. Since this button is easily accessible it would take no extra effort for the presenter to use this toggle switch, and with some indication on the screen that that mode has been entered there would be reduced error of accidentally entering this mode.

[User reports: When a user was asked “Is there a specific aspect of the application which you would like most to see improved?” they responded “I want to be able to write other notes on the screen”]

<u>Issue 3</u>

Moving around locks on the screen is difficult to do with the pen

Tasks two and three had the user use locks on the threads to illustrate issues in concurrency.

Problem 1

Users missed the locks when they were attempting to drag them to a thread because of their small size and possible very slight screen calibration issues. This caused extra lines being drawn on the screen which needed to be cleaned up by the user later.

Recommendations

There are two options for fixing this. The first is to make the lock icons larger so that they are more easily dragged. The second is to expand the area around the locks which are acceptable areas to drag the lock from. The second may be more aesthetically pleasing since the locks should not be overwhelmingly large in size when on the threads.

<u>Issue 4</u>

Simulation control options functions are not obvious

Tasks two and three asked the user to run a simulation of their completed diagrams.

Problem 1

Users expressed confusion about where the simulation controls were or what they were for. The plain grey button with the play icon in it did not draw the users eye properly.

Recommendations

This button should be more pronounced – using a different color and design. There also should be a label which makes it obvious that these are the simulation controls.

[User response: When asked if the interface was natural, a participant responded that they “didn’t know how to start the simulation”]

Conclusion

This usability report has provided an important insight into how users use an interface which has multimodal elements and what is required to make such a system successful. In any interface which intends to be naturalistic but also take advantage of the benefits of 21st century computing has a very careful balance to maintain between helpfulness of the assistance and the level of intrusiveness of the interface. This balance in the TabconV2 application has largely to do with anticipating and accounting for the different styles of handwriting, speech, and teaching being employed by the users of the software.

TabconV2 has been fairly successful in maintaining this balance although there are a few very important issues which need to be addressed. At least one of these issues, the issue of somehow differentiating annotations from the rest of the diagram, is not one which has been handled yet in any sketch based interface discussed here.

In contrast to speech assisted systems such as ASIST, TabconV2 does not force the user to speak in a specific way or use specific terms to use the system successfully, but uses this data if it is available to assist in recognition. There is still much work to be done in this area though. While the work presented here provides a certain proof of concept that a system such as this can improve recognition it does not run deep enough in the system to present an improvement which is extremely noticeable by the user in all situations. Currently this assists in recognizing shapes in the sketch, but helping recognize text and numerals written in the diagram could provide a marked improvement in usability.

The speech recognition does serve to reduce the errors in the diagrams, but it is not a substitute for a sketching system which allows for the diverse sketching styles of the intended audience of this software. This is an issue dealt with in all of the sketching systems that have been mentioned – they have a set of sketch elements which are valid which need to look a certain way once they are drawn to be recognized. For example SketchUML has very specific guidelines about where strokes connecting boxes must begin and end for them to work properly. This same issue affects Tabcon – a distinct second stroke must intersect the line portion of a thread and be recognized as a right or left chevron for the thread to be recognized. The rules need to become lenient enough so that users of all types can create drawings without training, but strict enough so that sketch elements are not misrecognized.

It has also been revealed from the usability study that the simulation controls could use some more explanation in the interface. The ASIST system uses a separate kinematic simulator (Working Model 2D) from the system which has been developed to recognize the sketch. Working Model 2D uses similar "play" and "stop" VCR style buttons to what is in Tabcon, but they label these buttons with their functions as well such as "Run", "Stop" and "Reset." Adding labels and colors to draw the user's eye to these buttons could enhance the usability of the simulation portion of the user interface.

Both Tabcon and TabconV2 are still in their infancy as research platforms and while they have started to become viable for use in the classroom for very specific applications there are still many issues to work out. This work has shown the viability of a multimodal speech/sketch interface for a classroom lecture type situation. With recognition of sketch elements approaching 90% the number of hurdles to make a system like this truly useful to a lecturer are becoming less and less.

Works Cited

Adler, A., & Davis, R. (2004). Speech and Sketching for Multimodal Design. *ACM IUI* , 214-216.

Adler, A., & Davis, R. (2007). Speech and Sketching: An Empirical Study of Multimodal Interaction. *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling.*

Alvarado, C. J. (n.d.). A Natural Sketching Environment: Bringing the Computer into Early Stages of Mechanical Design (Masters Thesis). 2000.

Apple Computer. (n.d.). *Speech: Recognizing Speech*. Retrieved from Apple.com: http://developer.apple.com/documentation/Cocoa/Conceptual/Speech/Articles/RecognizeSpeech.html

Apte, A., Vo, V., & Kimura, T. D. (1993). Recognizing Multistroke Geometric Shapes: An Experimental Evaluation. *UIST'93* , 121-128.

Corey, P., & Hammond, T. (2008). GLADDER: Combining Gesture and Geometric Sketch Recognition. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* , 1788-1789.

Dragon Systems. (n.d.). *History of Speech Recognition*. Retrieved from Dragon Naturally Speaking: http://www.dragon-medical-transcription.com/historyspeechrecognition.html

Gray, E. (1888). *Patent No. 386,815.* United States.

Hammond, T., & Davis, R. (2005). LADDER, a sketching language for user interface developers. *Computers & Graphics* , 518-532.

History of Voice Recognition Technology, The. (2004). *Information Management Journal* .

Hong, J., Landay, J., Long, A. C., & Mankoff, J. (2002). Sketch Recognizers from the End-User's, the Designer's, and the Programmer's Perspective. *Sketch Understanding, Papers from the AAAI Spring Symposium* .

Jurafsky, D., & Martin, J. (2000). *Speech and Language Processing.* Prentice Hall.

Kloonigames. (n.d.). Retrieved from Crayon Physics Deluxe: http://www.crayonphysics.com/

Microsoft Corporation. (n.d.). *Tablet PC*. Retrieved from Microsoft.com: http://www.microsoft.com/windowsxp/tabletpc/default.mspx

Minsky, M. R. (1984). Manipulating Simulated Objects with Real-world Gestures using a Force and Position Sensitive Screen. *Computer Graphics , 18* (4), 195-203.

Norman, D. A. (1988). *The Design of Everyday Things.* Basic Books.

Nuance. (n.d.). *Dragon NaturallySpeaking 10 Preferred*. Retrieved from Dragon NaturallySpeaking Solutions: http://nuance.com/naturallyspeaking/products/preferred.asp

Oviatt, S. (2000). Taming Recognition Errors with a Multimodal Interface. *Communications of the ACM , 43* (9), 45-51.

Qiu, L. (2007). SketchUML: The Design of a Sketch-based Tool for UML Class Diagrams. *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, (pp. 986-994). Chesapeke.

Qiu, L. (2007). *Tabcon: Tablet-based Tools for Teaching Concurrent Programming*. Retrieved from http://www.cs.oswego.edu/~lqiu/tabcon/

Rubine, D. H. (1991, December). The Automatic Recognition of Gestures (PhD Thesis).

Rubine, D. (1991). Specifying Gestures by Example. *Computer Graphics , 25* (4), 329-337.

Sarter, N. B. (2006). Multimodal information presentation: Design guidance and research challenges. *International Journal of Industrial Ergonomics , 36*, 439-445.

Schlegel, D. R., & Tenbergen, B. (n.d.). Naturalistic and Minimalistic Interfaces. *In Progress* .

Smith, T. (2007, July 19). *The IBM Thinkpad: 15 Years Old Today*. Retrieved June 28, 2009, from Register Hardware: http://www.reghardware.co.uk/2007/07/19/forgotten_tech_ibm_thinkpad/page2.html

Southerland, I. E. (1963, January). Sketchpad, A Man-Made Graphical Communcation System (PhD Thesis). University of Cambridge.

Sun Microsytems. (n.d.). *Java Speech API*. Retrieved June 30, 2009, from Sun Developer Network: http://java.sun.com/products/java-media/speech/