The Sensing and Measurement of Frustration with Computers

Carson Jonathan Reynolds
B. S. Technical Communication
University of Washington
May 1999

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in Partial Fulfillment for the Requirements for the Degree of

Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology

May 2001

Signature of Author: **Carson J. Reynolds**
Program of Media Arts and Sciences
May 11, 2001

Certified by: **Rosalind W. Picard**
Associate Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by: **Stephen A. Benton**
Chair, Committee for Graduate Students
Program of Media Arts and Sciences

The Sensing and Measurement of Frustration with Computers

Carson Jonathan Reynolds

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning, on May 14, 2001,
in Partial Fulfillment for the Requirements for the Degree of
Master of Science in Media Arts and Sciences

Abstract

By giving users a way to vent, we transform their frustration into a valuable source of information for adapting interfaces. Drawing from psychophysiology and tactile sensing, we present frustration sensors as a way of incorporating user feedback into interface design processes. This thesis documents the development of designs for several sensors aimed at detecting user frustration with computers. Additionally the thesis explores the design space between active sensors that facilitate the communication of frustration and passive sensors that detect frustration without demanding the user's attention. During evaluations we learned several things:

- Participants liked having devices to communicate frustration.
- The data that was collected during active and passive user interactions can be used for redesigning and adapting systems (either by hand, or automatically).
- User behaved differently during usability problems.

In a comparative study of three active designs (Frustrometer, Squeezemouse, and traditional feedback web page) we found that users prefer the Frustrometer to a web feedback page. Preliminary results suggest that frustration-stimulated behavior can also be detected through passive sensors. When combined with other contextual information, these sensors provide a crucial building block in systems that interact and adapt to human behavior by indicating where and when change is needed.

Thesis Supervisor: Rosalind Picard
Title: Associate Professor of Media Arts and Sciences

The Sensing and Measurement of Frustration with Computers

Carson Jonathan Reynolds

The following people served as readers for this thesis:

Hiroshi Ishii
Associate Professor
MIT Media Laboratory

Ted Selker
Associate Professor
MIT Media Laboratory

Acknowledgements

To begin with I'd like to thank all of the folks who helped directly with this thesis. My readers, Hiroshi Ishii and Ted Selker, have been generous with their time and patience. Raul Fernandez, Jocelyn Scheirer, and Mike Ananny provided invaluable council at various points during this work. Matt Norwood and Ali Rahimi provided intellectual and moral support for this work. Daryl Carlson deserves more thanks than I can give for all of his time and endurance.

In addition, all of the undergraduates and graduate researchers who assisted with this work also deserve special thanks. UROPs: Regina Sam, Sataporn Pornpromlikit, Maik Flanagin, Ian Finn, David Alexander, and Lindsey Wolf. Grads: All of the Affective Computing Group, Sumit Basu, Brian Clarkson, Rich Duvall, Ari Benbasat, Rob Poor, Craig Wisneski, Golan Levin, Casey Reas, and Elizabeth Sylvan.

I also need to thank the good friends who buoy me: Marshall Smith, Courtney Humphries, Dan Johnson, Lisa Brewer, Roger Squire, Michelle Peterson, Eli Smith, James Patten, Ben Piper, Paul Nemirovsky, Dimitris Vyzovitis, and Susan Pearlman.

David Farkas deserves thanks for believing in me enough to get me here.

Lastly, countless thanks to Roz for believing in me enough to bring and keep me here.

Love to my family for just believing in me enough.

This thesis is dedicated to walking out the front door.

Table of Contents

1. Introduction 11

- 1.1 Long Term Scenario - Improvisation
- 1.2 Short Term Scenario - Usability Tool
- 1.3 Motivation
- 1.4 Theory
  - 1.4.1 What is Frustration?
  - 1.4.2 What are Sensors?
- 1.5 Affective Interaction
  - 1.5.1 I'm going to teach you a lesson!
- 1.6 Contributions of the Thesis
- 1.7 Roadmap

2. Related Work 19

- 2.1 Psychophysiology
- 2.2 Tactile Sensing in Robotics
- 2.3 Tangible User Interfaces
- 2.4 Devices
  - 2.4.1 Sentograph
  - 2.4.2 Sheriden's Instrumented Classrooms
  - 2.4.3 Touch Phone
  - 2.4.4 Many Mice
- 2.5 Software
  - 2.5.1 Remote Evaluation
  - 2.5.2 Interface Instrumentation

3. Sensor Design 27

- 3.1 Preliminary Investigations – Ethnography
  - 3.1.1 Investigation Methods
- 3.2 Early Prototypes
  - 3.2.1 AffQuake
  - 3.2.2 iRX A/D Hack
  - 3.2.3 Beanbag
  - 3.2.4 First Mouse
- 3.3 First Revisions
  - 3.3.1 Single Force Sensitive Resistor
  - 3.3.2 Strip Chart Interface
  - 3.3.3 Yelling Detector
  - 3.3.4 Thumbs-Up / Thumbs-Down
  - 3.3.5 Bayesian Network

3.3.6 Voodoo Doll
3.3.7 Blink Detector
3.3.8 Expletive Detector
3.4 Second Revisions
3.4.1 Serial Swiss Army Board
3.4.2 Integrating Squeezemouse
3.4.3 FSR Positioning Inquiry
3.4.4 User Interfaces
3.5 Third Revisions
3.5.1 Attentional Considerations
3.5.2 Continuous Capture
3.5.3 Unobtrusive Mouse
3.5.4 Grid-SqueezeMouse
3.5.5 PressureMouse

4. Evaluation 49
4.1 First Pilot Study
4.1.1 Methodology
4.1.2 Preliminary Results
4.1.3 Analysis
4.2 Second Pilot Study
4.2.1 Anecdotal Observations
4.3 Second Study
4.3.1 Apparatus
4.3.2 Methodology
4.3.3 Results
4.3.4 Analysis
4.3.5 Analysis Conclusions

5. Applications and Future Work 65
5.1 Distributed Usability
5.2 Interface for Reinforcement Learner
5.3 Front End to Adaptive Generative Systems
5.4 Usability Benchmark
5.5 Contextual Fusion
5.6 Further Refinements

6. Summary and Conclusions 69

1 Introduction

You sit in front of your computer, absolutely livid. It is hours before an important deadline, and your word processor keeps doing all sorts of obnoxious things. When you enter in some new text, all of your diagrams are shifted out of place. All of the automatic features that usually seem pretty handy are causing you no small headache. And then it happens. The application crashes, taking your document with it. This is the straw that breaks the camel's back: you start yelling. Flush with anger and viscerally upset you start physically whacking the keyboard of your system. You start moving the mouse in an angry manner, as if it were not an inanimate object, but some appendage of a very problematic and naughty animal. Then you sit back and fume.

Figure 1. Displeased users sometimes physically vent frustration with computers.

A sympathetic friend sees your expression and inquires, "What's wrong?"

"Just the usual," You say sardonically. "Everything is fouled up and I don't have time to alter all the bleeding settings so that this thing behaves properly." The passerby nods knowingly and feels bad, wishing that there was something that could be done for you.

Imagine instead that the computer you are using is as sensitive, and emotionally intelligent as your friend. What would interaction be like if a computer had the capacity to detect and respond intelligently to your expressions of frustration?

1.1 Long Term Scenario - Improvisation

You are amused; the computer has finally gotten it. After a few nasty utterances of dissatisfaction with a user interface annoyance, the system has rooted out the problem and replaced it. As you sit in front of two screens, the system has improvised a new interface configuration on the mirror image of your current workspace. You shift your attention to the new improvised 'ping' to your 'pong.'

Figure 2. Pleasing users should be part of the design of computer interfaces.

Like an attentive and well-trained dog, your computer notices the satisfied sound of your voice and tracking your admiring gaze, elects to reinforce the improvised behavior. After a honeymoon phase with the new designed behavior has passed, the system incorporates the new design. Your computer captures the now unused workspace you previously attended to as a new canvas for improvisation. Random walks, various mutations of your current settings are displayed.

Like a complex diploid creature, an enormous space of different possible behaviors is encoded in each of the workspaces. The system stochastically explores new possibilities by looking at what you have liked and disliked in the past. When a particularly good candidate comes along, it's applied to your unused screen to see how you respond. And in this way you gradually tailor a previously one-size-fits-all interface to your own likes and dislikes.

1.2 Short Term Scenario - Usability Tool

You are not amused. Every time you use this particular feature you experience a good dose of aggravation. You aren't alone in your distaste either. A histogram hosted at the open-source project's development site shows that many other people have griped about this software's implementation.

A week later, you are content. Developers associated with the project made note of the spike on the histogram showing user dissatisfaction with the feature. Several on-line volunteers, fearing for their reputation as designers scramble to implement a better user-interface. The resulting patch, which you download and install, fixes the most menacing problems. Soon thereafter the developers on the project are happy to note that the spike

of frustration has receded, and focus their attention on the creation of other new and soon to be evaluated features.

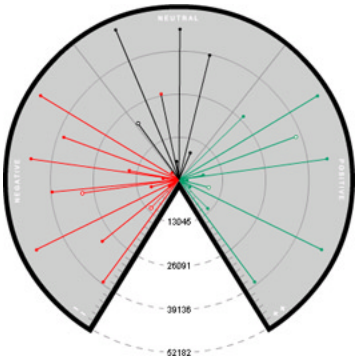


Figure 3. OpinionLab's interfaces visualize user satisfaction.

1.3 Motivation

For all the leaps and bounds computer science has made in the last few years, computers are still extremely clumsy and inelegant machines. To add insult to injury an increasingly computerized workplace and society necessitate interacting with computers on a very frequent basis. Usability engineers have tried to remedy many of the shortcoming and inelegant flaws that cause aggravation and irritation. But current usability practices seem unable to keep pace with the rapid pace of computerization. Traditional usability also seems poorly equipped to handle problems that stem from generic one-size fits all software.

Right now there is a large cost (in terms of time) associated with expressing feedback about shortcomings in a computer system. Typically, the only outlet available is to submit a comment on a software company's website. Before we can drastically improve computer interfaces, we must know what's wrong with them. But before we can do that, we need to make it easier for people to express their likes and dislikes. This thesis looks into interfaces and sensors that allow people to easily express their likes and dislikes about computer systems.

Once people are given tools to easily express what they dislike (or like) about their computer, all sorts of interesting possibilities emerge. With good data about the user's preferences, systems can begin to adapt the user interface to better fit.

Of course adaptation is no small matter. Consistency is held up to be one of the golden maxims of user interface design. Adaptation in many ways runs directly in the face of it. A system that constantly changes its behavior will be infinitely flexible, but infinitely irritating as well. Unfortunately the converse is equally disastrous. We have all experienced systems that are difficult or impossible to alter and often place the burden for change on the user. Given two extremes, amorphous inconsistency on one side, and stagnant uniform-like inflexibility on the other, what middle ground are we left to stand on?

Figure 4. Consistent, but generic interfaces, may not allow enough flexibility to ease frustrations.

Reeves and Nass argued in *The Media Equation* that we form social relationships with mediating technologies like computers. [1996].

The space is clearly staked out when we think about the set of circumstances when people would not only accept, but also desire a change of behavior. In some sense, we are already equipped with tools for training other creatures how we like to work. When we grow frustrated or angry at a pet, child, or friend we are often trying to get them to alter their behavior. More specifically, to feel 'bad' about something they've done, and to sincerely try to avoid doing it again.

Likewise, we can say that when someone is frustrated with their computer, they are interested in changing its behavior to help them achieve the user's ends instead of thwarting them. So if we can reliably detect and measure frustration, then we also have good indicators about where, when, and why we should adapt.

1.4 Theory

1.4.1 What is Frustration?

When early psychological experimenters began to do behaviorist studies, they began to more rigorously define the notion of frustration. Psychologists came to see frustration as the negative feelings that arise when attempts to achieve a goal are thwarted.

1.4.1.1 Frustration's Distinction from Stress

We may be tempted to lump frustration in with the notion of stress. The two are often used interchangeably. However there is an important distinction to be struck between the two. Stress arising from heavy mental, physical, or emotional workloads does not necessarily coincide with frustration.

Briefly, someone may be stressed without being frustrated. Conversely, someone may be frustrated, but not mentally or physically stressed.

1.4.1.2 Frustration's Expression

Frustration can be expressed in many ways. We may communicate frustration to others through our posture, facial expression, or tone of voice. Our autonomic nervous system may respond to frustration by increasing skin conductivity, decreasing heart rate variability, or tensing muscles. Cognitively, we

may verbalize or alter our behavior in response to frustration. In this thesis we explore sensors for detecting physiological changes (like skin conductivity or muscle tension) and behavioral changes (like verbal expressing or physically venting frustration).

1.4.2 What are Sensors?

If we are about to set about designing sensors, we may first want to ask ourselves precisely what entails the notion of a sensor. A sensor is a device that measures some observable physical quantity.

Sensors are often transducers meaning that they convert a physically measurable quantity (like temperature, or pressure) into an alternative signal. A microphone, we could say, transduces changes in pressure (vibrations) to a change in electrical signal.

1.4.2.1 Active vs. Passive Sensors

What is curious is that sensors are often used as interfaces. A mouse, the most common pointed device used with computers, is actually a two-dimensional motion sensor. But it is used as a way to manipulate information on computers.

When a user is actively aware of a sensor and interacts with it in a way that requires some of their attention, it becomes an "active" sensor, something with which people interact. When a user is not consciously aware of a transducer, it is more "passive" meaning that it performs more traditionally as a sensor that monitors, without requiring the attention of the user. Generally, the more active the sensor, the more mental or physical effort the user has to apply to use it. Passive sensors require less effort on the part of the user and may be used while the user is engaged in another task.

Much of the work presented later lies in the surprisingly abundant space between passive and active sensors. Following the experience of designing and evaluating several devices that are hybrid sensor-interfaces, we are able to say some concrete things about design trade-offs in this space.

1.5 Affective Interaction

Our communication with computers is impoverished. The command line interface lets us say what we want the computer to do. However, there are very few interaction styles that let me express how I like what the computer did. If I am frustrated by a friend's behavior, more often than not, they can read my expression and alter their behavior, even before I consciously say "Could you not do that, I find it annoying." But even the 'smartest' adaptive interfaces do not even begin to offer this sort of emotional intelligence.

1.5.1 I'm Going to Teach You a Lesson!

In some sense, we already have developed behavioral semaphores that we use to teach each other how to interact. When I am pleased with what a friend has done, I will smile and convey my happiness with them. When I am angry about what my dog has done, I may scold the dog, say "Bad Dog" loudly. A child learning is filled with a world of good and bad expressions. These expressions are gradually learned, and favorable behavior is gradually reinforced, while distasteful behavior is gradually learned to be avoided. However, when we look at most adaptive interfaces today, I cannot easily train them to avoid behavior that I dislike and to adopt behavior that I find pleasing. Passive or active sensors that can help communicate my affect may allow for a more interesting sort of interaction.

1.6 Contributions of the Thesis

First and foremost, this thesis documents the design of several sensors developed as tools for the exploration of communication of user feedback related to frustration. The document also seeks to provide information about the interplay between sensors on one hand, and interfaces on the other. Lastly, this thesis seeks to show that unobtrusive sensors can be used to distinguish between frustrated behavior and its converse.

1.7 Roadmap

The next chapter concerns itself with documenting previous and ongoing related work (chapter 2). Following this is a presentation of the design process of various sensors and interfaces for the expression of frustration (chapter 3). A series of

studies which evaluate these devices and interfaces is included immediately afterwards (chapter 4). The conclusions of data analysis can also be found in this chapter. Lastly, various applications and future work concerning sensors that can detect frustration are outlined (chapter 5).

2 Related Work

The path to make user interfaces more adaptive and in step with more natural modes of human interaction is a long and twisting one. Many sub-fields and notions must be borrowed from in order to make meaningful progress.

2.1 Psychophysiology

Human factors engineers have long concerned themselves with the quantification of the physiological and psychological effects of different machinery. Psychophysiology seeks to find biological signals or behavior that are linked to stress (among other states). Lie detector tests, and an extensive amount of work on workplace satisfaction are rooted in the inquiries of psychophysiology.

Previously, engineers had used different physiological measures of strain to assess designs [Backs and Boucsein, 1999]. But more recently, Picard suggested that "frustration could be measured, quantified, and incorporated into the evaluation of new products" [Picard, 1997]. The sensing of affect, or emotional expression, provides a mechanism for triggering adaptation. Affect sensing focuses on the measurement of signals related to the expression of frustration, and emotion in general.

Work in the affective computing group and elsewhere has shown that different affective states can begin to be distinguished by computers. For instance, a 24-subject experiment was run and galvanic skin response, blood-volume pressure, and mouse-click behavior were recorded from test subjects who were intentionally frustrated. The resulting data set could be recognized by a classification program with 67.4% accuracy [Scheirer et al., 2001].

Figure 5. Face classified by CMU's Automated Facial Expression Analysis system.

Another approach that does not use biosignals, but instead builds upon computer vision, is the classification of facial expression using Paul Ekman's Facial Action Coding System [1978]. Work done at Carnegie-Mellon University's Affect Analysis group was able to correctly classify different facial expressions with accuracy rates greater than 90% [Tian et al., 2001].

The obvious next question is that if we can begin to detect affective states—like frustration—with the aid of electrodes and cameras pointed at the user's face, can we do the same with less obtrusive sensing technologies? For instance can we determine someone's stance toward a computer by examining touch patterns?

2.2 Tactile Sensing in Robotics

Robotics offers us not really an answer to this question, but some implements with which to explore it. The field of tactile sensing in robotics concerns itself with making mechanical and electrical transducers for touch. If truly embodied robots are to be realized, clearly they need to be able to grasp and touch. But they may also need to carefully run their fingers over, or maybe even caress.

Starting with the simplest sort of information, a single bit (touching / not touching), sensors have gradually expanded the range and resolution of the touch that can be transduced. A one-bit touch sensor has become a simple matter to produce. We can use mechanical switches, a photodiode, and electrodes (for conductive surfaces). For instance, older supermarkets made use of one-bit touch sensors to open doors. Many robots have used antenna-like feelers to avoid repeatedly bumping into things.

Figure 6. The MIT Artificial Intelligence Lab's Genghis robot uses antennas as tactile sensors.

One-dimensional sensors provide a degree of information instead of a single bit. Various force sensitive resistors and capacitive sensors have been used widely in this capacity: not simply to determine if something is touching or not touching but also to determine how hard something is touching.

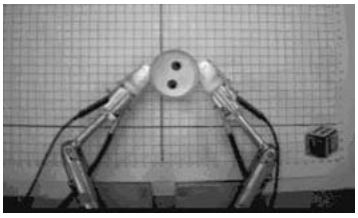


Figure 7. The Harvard Tactile Sensing Project's tactile fingers make use of video flow analysis to sense touch.

Crowder provides an excellent survey of different tactile sensors that have been used in automation and robotics [1998]. He taxonomizes tactile sensing by dividing it into touch, spatial-tactile, and slip sensing. He also enumerates different techniques that can be used to build touch sensors:

- Mechanical
    - Load
    - Pressure
- Electrical Measurement
    - Resistive
    - Capacitive
    - Magnetic
- Optical Measurement
    - Photoelastic
    - Fiber optic

Most recently, researchers have been working to develop deformable arrays of tactile sensors to coat surfaces. Nicolson and Fearing have developed non-conformable planar and cylindrical tactile sensors that are a strong step in this direction [2001]. Perhaps the best example so far of a conformable sensor is Pressure Profile Systems' 8x8 conformable array.

2.3 Tangible User Interfaces

Closely related to this endeavor is the exploration of haptic interfaces to computation and the work of the Tangible Media group at the MIT Media Lab. The group's early work envisioned giving users the ability "to 'grasp & manipulate' bits . . . by coupling the bits with everyday physical objects and architectural surfaces"[Ishii and Ullmer, 1997]. The Tangible Media Group's Phicons are a concatenation of "Physical" and "Icons" meaning graspable representations for software bits. This idea was developed into the notion of a feedback device for affect by Matt Norwood's "thumbs-up" and "thumbs-down" interface for Mr. Java, an enhanced coffee machine [2000].

2.4 Devices

Communicating expression is actually a very old pursuit in many senses. Almost every musical instrument could be said to perform in this capacity. Instruments act as a translator between physical manipulations and the communication of some sound, which often conveys some mood. Very recently, people begun to think of these physical manipulations as something that could be quantified and distinguished between.

Figure 8. The Clynes Sentograph.

2.4.1 Sentograph

Manfred Clynes built an artifact for expressing emotion, inspired by the breadth of expression distinguishable in massage via touch [1977].

His sentograph allowed a user's touch to be measured and visualized along a two-dimensional space. By categorizing the resulting shapes, Clynes was able to distinguish between eight fundamental classes of tactile emotion expression: anger, hate, grief, joy, love, romantic love, reverence, no emotion. Clynes found that common expression of these shapes exists across cultures.

2.4.2 Sheriden's Instrumented Classrooms

Another early attempt of communicating expression through the use of artifacts had quite a different end-goal. Twenty-five years ago, Sheridan discussed instrumentation of various public speaking locations (classrooms, council meetings, etc..) with switches so that the audience could respond and direct discussion by voting [1975]. If a particular topic was of interest to the majority of the audience, they could flip physical switches to express this preference.

2.4.3 Touch Phone

Jocelyn Scheirer's touch phone is another instance of a tangible interface for expressing emotion [Scheirer and Picard 2000]. It converts grip strength sensed on a conductive foam surface into a colored representation on a computer. It is used to communicate expressive pressure changes to other people (as opposed to the computer itself).

2.4.4 Many Mice

Several different people have stumbled on the idea of using the mouse as medium for sensing affect related to computers. Various approaches have been tried, from examining mouse movement behavior [see Mueller and Lockerd, 2001] to embedding electrodes onto a mouse. Each of the following mice represents a tangible interface that can be used to collect information about the user's affect.

2.4.4.1 Sentic Mouse

Kirsch developed a mouse modeled after Manfred Clynes' sentograph [1997]. The mouse made use of a force-resistor to sense directional input. Kirsch used this mouse as part of study in which participants were shown imagery from Lang's International Picture Affective System.

Figure 9. Kirsch's Sentic Mouse.

2.4.4.2 Touch Mouse

Borrowing aspects of the Touch Phone design, Norwood implemented a mouse that could detect how hard someone was squeezing around a line of conductive foam. Using a data acquisition board, this signal was passed into a computer and visualized using a color swatch in a similar manner to the Touch Phone.

2.4.4.3 Squeekee

Squeekee is a mouse that detects how hard users have clicked its buttons. A commercial product, it seems to be targeted at games, but its inventors are also interested in other innovative HCI applications. Although it is not used explicitly to detect affect, it provides a dimension of information about clicking; it seems a likely candidate device that could be used to detect the expression of frustration.

Figure 10. Squeekee detects how hard users click.

2.4.4.4 MS Touch Mouse

Microsoft's Ken Hinckley developed a touch sensitive mouse that relayed information back to the user's interface [1999]. The mouse only conveyed a single bit of information: whether the user was touching the mouse or not. This bit in turn was used to determine whether application toolbars should be made available. It is perhaps one of the simplest, but better realized examples of a perceptual user interface.

Figure 11. The MS Touch mouse makes application toolbars visible when touch is detected.

Figure 12. The IBM Emotion mouse collects temperature, galvanic skin response, and somatic movement.

2.4.4.5 IBM Emotion Mouse

A comparatively elaborate approach is being pursued by researchers at IBM. The emotion mouse developed in their lab is used to gather temperature, galvanic skin response, and somatic movement from computer users. Coupled with heart rate information that is acquired from a separate chest-strap sensor, researchers used the mouse to distinguish between six different emotions: anger, fear, sadness, disgust, happiness, and surprise. Preliminary results suggest that they were able to achieve 66% recognition accuracy rates using this device [Ark et al., 1999].

2.5 Software

Usability researchers have also pursued the idea of using software interfaces to communicate affective information about computers. The simplest manifestation of this idea is the nearly ubiquitous feedback web page. As crude as comment textboxes are, they are an important step towards facilitating the communication of frustration. In some sense they convey parts of the usability lab or marketing department directly into the use environment. This idea is developed more fully by the notion of remote evaluation.

2.5.1 Remote Evaluation

Remote evaluation, put simply, is using communication systems to collect usability information over a distance. Castillo et al. introduced the notion of remote evaluation through the use of critical incident reporting tools [1997]. More specifically, Hartson et al. suggest that remote tools can be especially useful when supporting "formative evaluation" after software has been released [1996]. But Castillo et al. also note that remote usability can be used during product development and as part of customer support. They suggest a myriad of different methods from the more mundane questionnaire submitted remotely up to video-conferencing and screen sharing software in support of evaluation.

Harston et al. performed a feasibility case study that "determined that user-reported critical incident method was a feasible method" for collecting usability data [1998]. Elsewhere they conclude that semi-instrumented remote evaluation tools could "provide approximately the same amount and value of qualitative data" as what would be obtained through more traditional usability laboratory based methods.

2.5.2 Interface Instrumentation

Interface instrumentation is the recording of interface usage data for analysis. Various research and commercial systems have been developed to explore the idea of instrumenting interfaces. For instance WinWhatWhere™ provides software for monitoring activity on remote systems [2000]. Reportedly, researchers at IBM have also developed live-motion screen capture software called UCDCam, based on Lotus ScreenCam™ [1998].

Swallow et al. argue that "thousands of users could be using instrumented applications"[1997]. They also note that automatically recorded information can be used to detect usability problems. They developed a set of problem indicators such as the invocation of on-line help, or triggering of an error message that can be monitored.

2.5.2.1 OpinionLab

One of the most developed and commercial successful remote evaluation tools is a small widget that is embedded in company's web pages. OpinionLab's animated [+/-] icon serves as a remote evaluation tools so that web site designers can focus their efforts on the more problematic parts of their website.

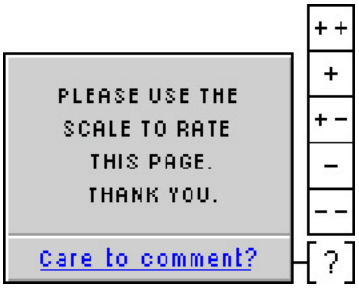


Figure 13. OpinionLab's remote evaluation interface.

3 Sensor Design

This chapter details the design process that was used to build upon the work documented in the related work chapter. It begins by explicating a participatory design process that influenced many of the later design decisions. Descriptions of the very first (but somewhat crude) prototypes that were developed follow. The remaining bulk of the chapter details iterative revision of these early prototypes into more developed and mature designs.

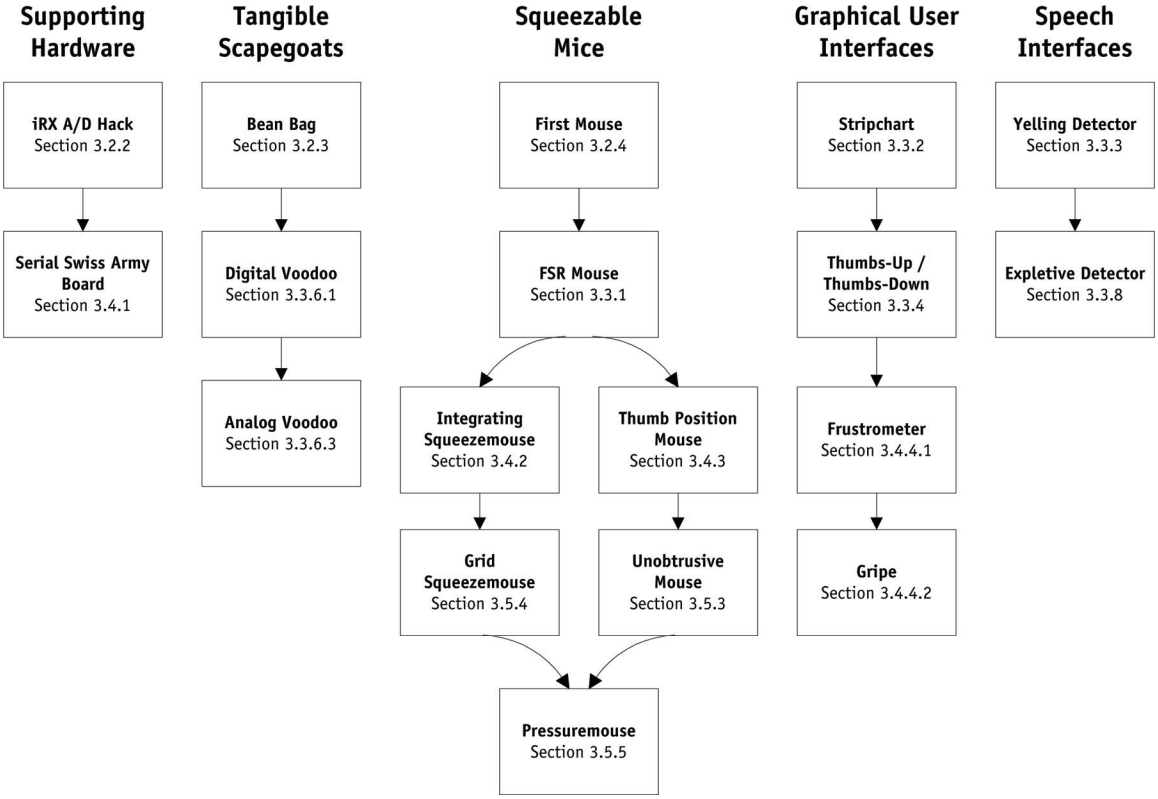


Figure 14: The chart above details the development of the different sensors explicated in this chapter. The successive levels of depth represent the number of iterations. A large breadth of sensors populates the space spanned by the notions of passive and active user interaction. The ecology of sensors was gradually culled down into the promising and more-fully realized prototypes.

3.1 Preliminary Investigations - Ethnography

This thesis' design process was motivated by an effort to understand how people might go about using adaptive systems. So the work of sensor design actually began as an contextual inquiry into how people might use adaptive systems.

Two driving questions were: How should people initiate and use adaptation? And how should adaptive systems that respond to frustration be represented? Much of the work that is presented further on is based on the findings of this inquiry.

A small pilot inquiry was performed to try to work through many of the theoretical and practical issues surrounding the powerful but nebulous notion of adaptation. Borrowing Holtzblatt and Beyer's concept of participatory and contextual design [1997], I chose to form a small group of two representative users with whom I could perform critical design reviews in which my collaborators would role-play scenarios with low-fidelity paper mock-ups.

3.1.1 Investigation Methods

During each iteration I would present a "thesis" about how they might use adaptation in the form of a paper mock-up. My collaborators were encouraged to redesign the interface to better suit their needs. This was considered the "anti-thesis". Then, we would form (and cognitively walk through) a "synthesis" which took the best attributes of the prototype and their response to it. This synthesis then served as the blueprint for the next low-fidelity mock-up that I created offline and was used as the "thesis" for the next interview. So the forming of the interface was an iterative, dialectical process of sorts. Think-aloud protocol, in combination with narrative storylines, was used to try to tease out the collaborators' opinions about particular interface metaphors and task structures.

3.1.2.1 Thesis 1: Accessible

Starting from Holtzblatt and Beyer's suggestion to "invent solutions grounded in user work practice" I began by thinking of ways to integrate adaptive capabilities into existing interface paradigms. After casually observing users, I noted that users typically searched menus looking for alternatives when they reached some sort of usability stumbling block that caused frustration. Based on this anecdotal evidence, I developed my first thesis about how adaptation ought to be integrated into the user's interface: it should be accessible.

More specifically, I decided that adaptive features should be as available as help menus, or perhaps should be considered part of help. Accordingly, I developed prototype mock-ups using Microsoft Visual J++:

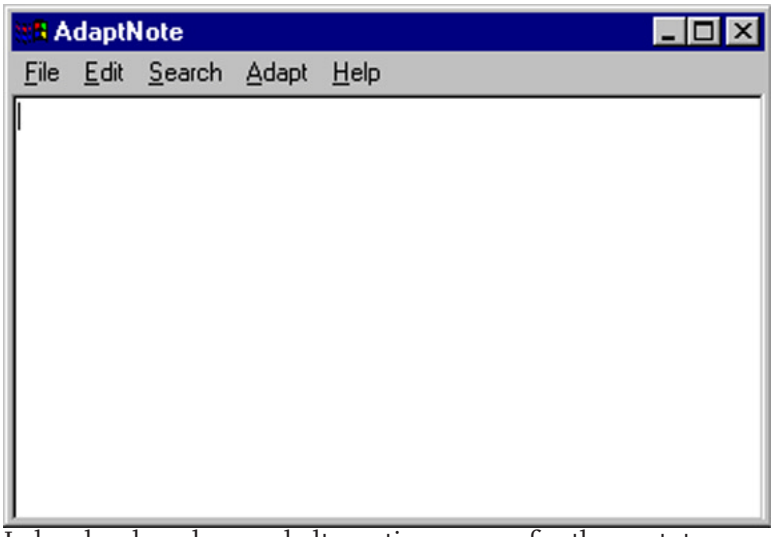


Figure 15: AdaptNote mock-up, with Adapt menu.

I also developed several alternative menus for the prototypes:

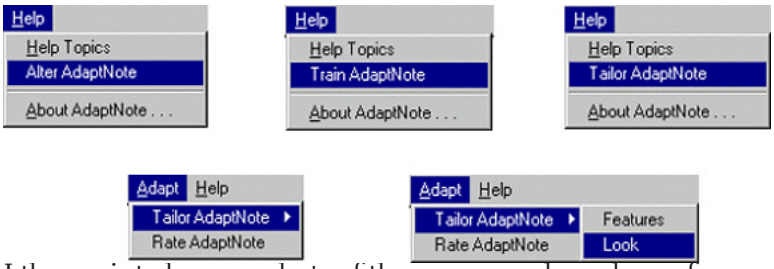


Figure 16: Various menu configurations for AdaptNote mock-up.

I then printed screen shots of the menus and mock-ups for use during my interviews. I encouraged participants to annotate these mock-ups and alter them in any ways they found useful. My collaborators made similar observations about this thesis:

- "Edit" seems to be the intuitive place to find tools to adapt the interface
- If adaptation would not be limited to just one application, then the tailoring tool should not be part of the program, but part of something "above" it like the operating system.
- The tool should not just be accessible but visible
- This prototype helps users who know what they want to get rid of; the interface should be more intelligent.

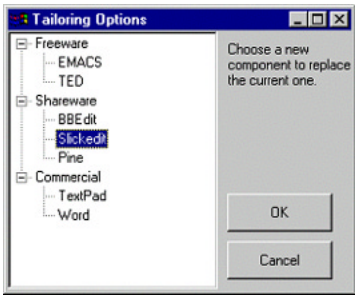


Figure 17: Dialog invoked by AdaptNote menus.

Using this mock-up as a springboard, the participants and I collaboratively developed a new storyboard for how action might go. This storyboard was the synthesis of the original prototype and their criticisms.

3.1.2.2 Thesis 2: Intelligent

The consensus seemed to be that the menus did not provide an adequate interface for adaptive features. Furthermore, the task analysis that I originally performed seemed inadequate. Consequently, a new interface and task structure was designed for my second round of interviews.

The second iteration focused on a method to intelligently help users of the system by suggesting alternatives. Since novice users seem to have a hard time locating features I reasoned that it would be better if the system could interrupt the user when they were sufficient frustrated and present an alternative. The second prototype consisted of a tailoring avatar and accompanying "wizard".

My collaborators responded very negatively to this mock-up of the interface. Their previous bad associations with Microsoft's Office Assistants ruled out the use of an avatar altogether. They felt that the avatar assumed a certain amount of childishness and lack of intelligence in users, which made them feel uncomfortable. Furthermore, they disliked the idea of a program continuously interrupting them. They preferred to be in control.

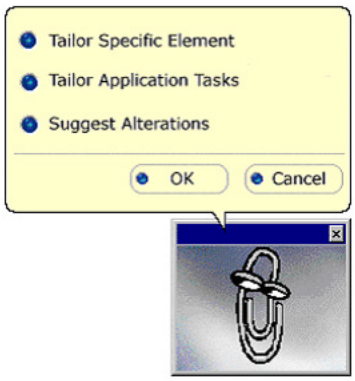


Figure 18: Office Assistant interface mock-up.

Clearly, this was not the vision of adaptive intelligence they were after. The whole experience was beautifully summed up by one participant: "There is a fine line between knowing you and trying to think for you!" Accordingly, my co-designers asked me to look for other ways of monitoring and responding to frustration that were not so obnoxious.

3.1.2.3 Thesis 3: Perceptual

At this point I was somewhat at a loss as to how to move ahead. Fortunately for me, a talk by Mathew Turk on perceptual computing provided me with some direction. I decided to redefine the task structure to include: continuous monitoring of frustration, intelligent suggestion of alternatives, and assisting in applying an alternative. From this I developed a somewhat unorthodox perceptual interface to help the computer monitor user frustration:

In my third round of interviews my collaborators found this

Figure 19: Conceptual imagery of perceptual interfaces for adaptive system.

model to be much better than the previous iteration. They did have some misgivings though, they had strong objections to the computer interrupting and suggesting alternatives, even if the interruption was in the form of dialogs instead of avatars; they wanted the interface to be less obtrusive. They also suggested that it would be beneficial if the system allowed them to experiment with and "undo" the various alternative components.

3.1.2.4 Thesis 4: Transparent

I synthesized the notes from my last interviews into one final thesis. It seems that the adaptive interface should follow a task structure that includes:

- Transparent monitoring of frustration
- User to control and initiate adaptation
- Intelligently assist user in finding alternatives
- Help user apply and undo various components

Of course, due to the extremely small number of participatory designers, these initial observations needed to be confirmed by building and evaluating more prototypes.

3.2 Early Prototypes

Starting from this initial inquiry, I began to focus my attention on building actual prototypes that facilitated the communication of frustration. The more promising of these were gradually refined into more elaborate and higher-fidelity prototypes.

3.2.1 AffQuake

One early notion of how a software interface might sense affect was AffQuake. As an application for the Galvactivator glove, AffQuake transduced galvanic skin responses and caused avatars within the game to respond to this signal [Picard and

Figure 20: AffQuake sized avatars in response to skin conductivity.

Figure 21: Jocelyn Scheirer's galvactivator communicates skin conductivity through a LED and a jack which analog data can be acquired through.

Scheirer, 2001].

In AffQuake, if the player became startled, (their skin conductivity sharply peaked) then their avatar within the game would also reflect the state of being startled by jumping back. A more elaborate version involved changing the size of characters within the game in relation to the level of arousal detected. The more excited a player got, the more their GSR signal would rise. The resultant signal was passed into quake causing the player's avatar to be scaled correspondingly.

AffQuake let me investigate one possible way in which someone's affect would alter interaction. I found that skin conductivity is convenient, because it can be acquired from the user's hand. However I also found this inconvenient because users had to plug in a cable, which sometimes interfered with typing. Furthermore, I observed that skin conductivity changes at a relatively slow rate, and lags slightly behind the inducing psychological stimulus. Consequently, real-time interaction that makes use of skin conductivity seems to be slow and unresponsive to end-users.

3.2.2 iRX A/D Hack

AffQuake required a data acquisition board. Most off-the-shelf PCI acquisition boards were expensive, and can only be installed in one computer. A much simpler sensing board that cost less and could be easily attached to computers was needed. After some inquiry, I was able to combine the work of the iRX project and the sensing technology used as part of the smart shoe project. More specifically, I altered the iRX board to use a PIC16C711, microcontroller, which incorporates an analog to digital converter.

3.2.3 Beanbag

Another early, and promising way in which people can communicate their frustration to the computer was through the use of a beanbag interface. In an effort to construct a tangible interface that people could physically manipulate we disassembled a mouse and placed its parts inside a beanbag. As the beanbag was squeezed or struck, the beans inside rolled against the mouse's wheels causing data to be sent. It was—in some sense—a one-bit touch sensor, able to determine if the beanbag was jostled.

Figure 22: Beanbag touch sensor.

The beanbag was tied in with simple monitoring which was able to relay squeeze messages to open applications. A simple version of Word was created which shut off the Paperclip office assistant when its proxy, the beanbag, was struck.

3.2.4 First Mouse

At the same time, I wanted to explore more traditional interfaces that could be augmented to transmit information about the user's affect to the computer. After examining Matt Norwood's Mouse (See section 2.4.5) I undertook the building of a squeeze sensitive mouse. One of the shortcomings of Norwood's mouse design was that it responded differently depending on where the user squeezed the mouse. I settled on a simplification that made only a single location on the mouse sensitive.

The sensor consisted of metal plates separated by conductive foam. As the plates were squeezed together by the user's tightening grip, the conductivity between the two plates increased.

The difficulty experienced with this design was that it was hard for the user to control. Its response was not easily predictable from the user's perspective. In addition, the dynamic range of the sensor was limited.

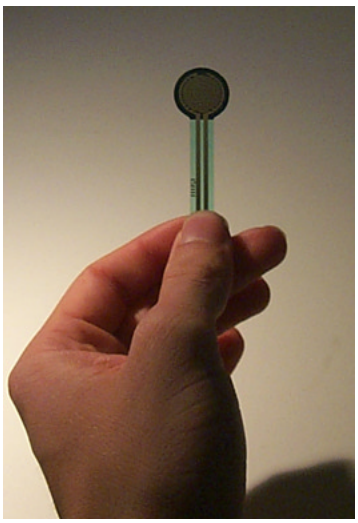


Figure 23: Force Sensitive Resistor.

3.3 First Revisions

These crude prototypes gave us some devices with which we could think critically about sensor design. Although these first versions had their limitations, they served as a stepping-stone to more complex designs.

3.3.1 Single Force Sensitive Resistor

In order to get a more uniform response, we turned our attention to sensors that transduce force. Interlink Electronic's force-sensitive resistors (FSRs) had previously been used in the creation of electronic musical instruments. Force sensitive resistors consist of a printed electrode pattern positioned over a conductive polymer

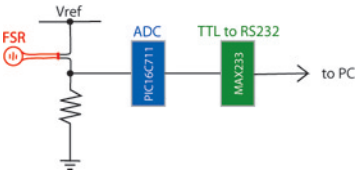


Figure 24: FSR block diagram.

We used the force sensitive resistor as a variable resistor in a circuit that we added to the iRX analog to digital converter. The FSR forms the top half of a voltage divider, which changes the amount of voltage passed to the ADC as the user squeezes harder. This design was beneficial because it was easy for the user to locate and use the pressure sensitive spot, and the response of the resistor to pressure was very predictable. What's more, the design was simple to construct and facilitated rapidly prototyping different designs. We were later able to build and compare many different arrangements for the FSR, like under the user's palm, closer to the heel of the hand, and beneath the user's thumb. (See section 3.4.3)

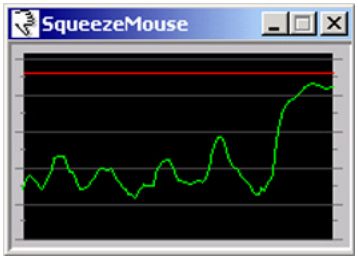


Figure 25: Strip Chart Interface.

3.3.2 Strip Chart Interface

We decided to provide feedback to the user about pressing on the mouse. Inspired by logging oscilloscopes we set about to visualize the mouse input as a plot on a strip chart graph. This was beneficial, because it allowed for users to see a short history of their input using the mouse. We added a threshold line so that users can see when they are squeezing very hard.

3.3.3 Yelling Detector

After starting with touch, I became interested in other modalities that people might use to express frustration. One idea was to build a simple yelling detector. Using a PC microphone I built an interface that detected the amplitude of a sound sampled from the microphone. Sounds over a certain amplitude were assumed to represent yelling.

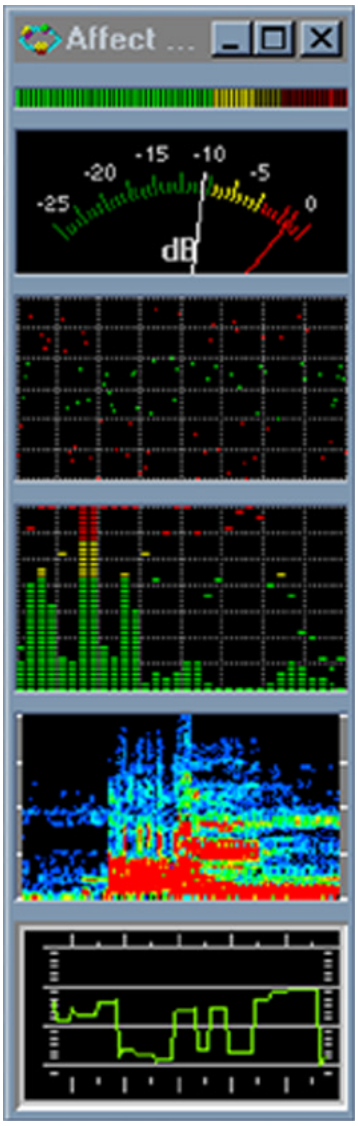


Figure 26: Yelling Detector interface.

3.3.4 Thumbs-Up / Thumbs-Down

Each of the interfaces created thus far mapped well to the notion of arousal, or the level of user excitation. However these interfaces were not good at allowing the communication of valence, whether the excitation is positive or negative.

To better provide for more sophisticated models for emotional behavior like Schlosberg's [1953], I began to explore interfaces that communicate valence. I designed and built thumbs-up and thumbs-down icons so that the user can register pleasure or displeasure with different interactions.

Figure 27: Thumbs-Up / Thumbs-Down

3.3.5 Bayesian Network

As part of a conceptual demonstration, the single FSR squeeze mouse was coupled with the yelling detector and thumbs-up

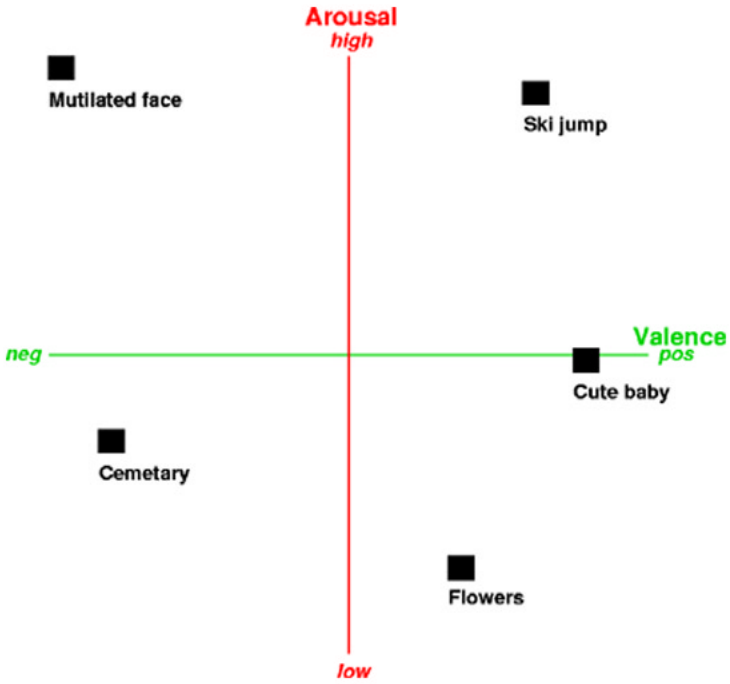


Figure 28: Schlosberg suggested arousal and valence as two primary dimensions of emotion.

/ thumbs-down, to provide indicators of the users frustration which were passed into a network to estimate the conditional probability. I collaborated with Jack Breese, who used the Microsoft Bayesian Network Toolkit to design the network graph. The output of this network was broadcast to applications. One demonstration application altered the conversational behavior of the Microsoft Office Assistant when the user was detected to be in a frustrated state. The Bayesian network provides a good framework for sensor fusion, when the inputs of the sensors may not be completely reliable.

3.3.6 Voodoo Doll

The Beanbag interface (section 3.2.2) initially seemed one of the more promising early prototypes. People liked to play with it, and the interaction mapped directly to a real world metaphor: the use of voodoo dolls.

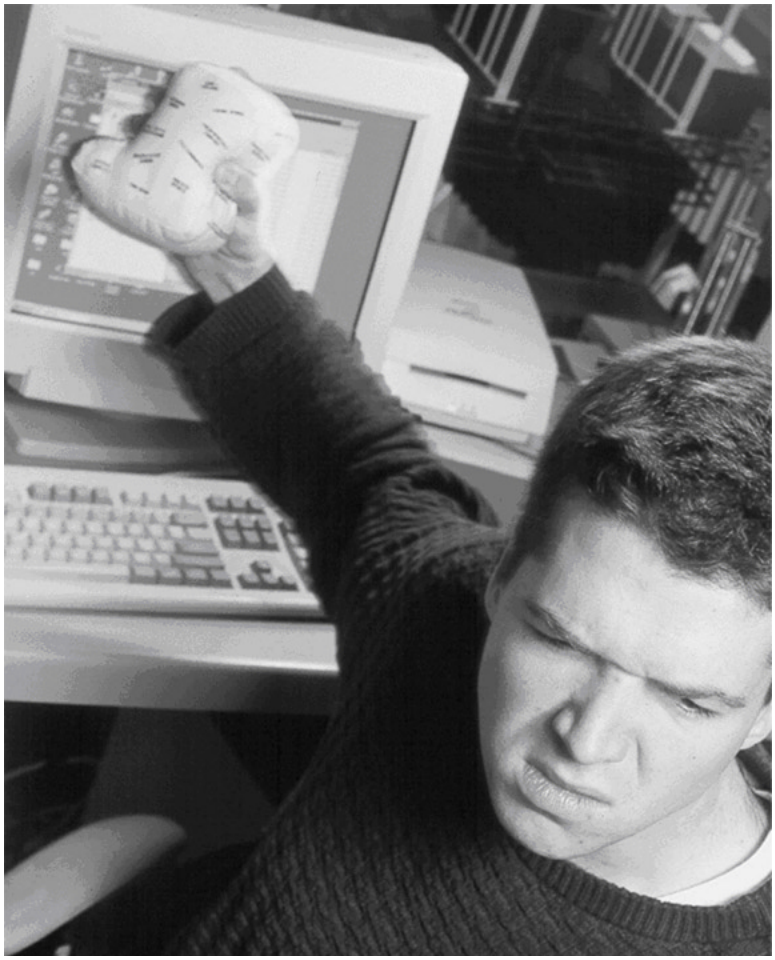


Figure 29: The Voodoo Doll in action.

But it had several strong shortcomings. The Beanbag was a one-bit touch sensor; it doesn't give any indication of the depth of the user's frustration. It was also cabled to the computer, limiting its use. The Beanbag could not easily be thrown. What's more, if the computer it was tethered to crashed then any sort of data collected would be lost as well.

With the help of several undergraduate researchers, but most notably Daryl Carlson, I set about to remedy these shortcomings. We chose to use Craig Wisneski's personal ambient display design as a jumping off point for our transmitter [1999]. The voodoo doll owes a strong debt to the design used by Wiseneski's personal ambient displays. Craig pioneered an architecture for transmitting RF information between base station and small ambient displays. Craig's Master's thesis envisioned people interacting with abstract information through tangible, subtle interaction. For instance, one personal ambient display relayed information about the user's stocks by growing hot when the market was up, and growing colder when the market was down.

Following Wisneski's design, we chose to divide (and conquer) the design into two components:

- A proxy (accelerometer and transmitter)
- A base station (receiver, microcontroller, and serial adapter)

We then proceeded by choosing a physical form factor. Archie McPhee's (www.mcphee.com) sells gag voodoo dolls for people who are frustrated with their computer. We figured this was an ideal starting point, since it already evoked the idea of punishing your computer.

We ordered several dolls and quickly turned to assessing different wireless radio transmitters. After looking at several radio frequency (RF) modules we settled on the HP-II series transmitter from Linx Technologies (www.linxtechnologies.com). We started with an evaluation kit that contained prototype boards and transmitter / receiver modules. We discovered that the transmitter board was much too large to fit inside the voodoo doll I wished to use. This meant that we had to set about developing our own transmitter board.

Figure 30: The original use of the Voodoo Doll.

3.3.6.1 Digital Voodoo

We began by examining prototypes of Ari Benbasat's [check spelling] gesture tracking accelerometer transmitter board. We chose to use the ADXL210AQC, a 2-axis accelerometer from Analog Devices that uses a pulse-width modulation scheme (http://www.analog.com/industry/iMEMS/products/ADXL210.html).

In order to use a checksum scheme, ensure line balancing, and avoid signal attenuation, we chose to embed a small microcontroller on the transmitter board. Because of our familiarity with the series, we selected Microchip's PIC microcontroller family (http://www.microchip.com). The PIC16F84 is a small 8-bit core that runs at 10 Mhz and is well suited for small digital decoding tasks.

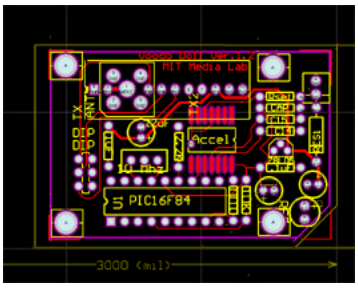


Figure 31: Voodoo PCB schematic.

We then constructed a preliminary PCB design using the ProTel circuit design package. The design was not very compact, but was still small enough to fit inside of the voodoo doll. This first revision worked correctly, but was large and a little unstable. Consequently, we made another revision. This second PCB was designed to be much more compact.

The second transmitter board consists of:

- Several capacitors for noise reduction
- An LED to indicate PIC status
- A 5-volt voltage regulator
- A 10 MHz oscillator
- A dip switch to select the transmitter frequency
- A PIC16C84 microcontroller
- A Linx transmitter module
- The transmitter's whip antenna

We used AP Circuits (www.apcircuits.com) as a board fabrication house. They had a two-day turn around which allowed for quick prototyping. AP provided us with printed circuit boards, which we soldered components to and tested. With the transmitter workably designed, we turned our attention to the base station next. Thankfully, it did not prove to be as difficult a design problem.

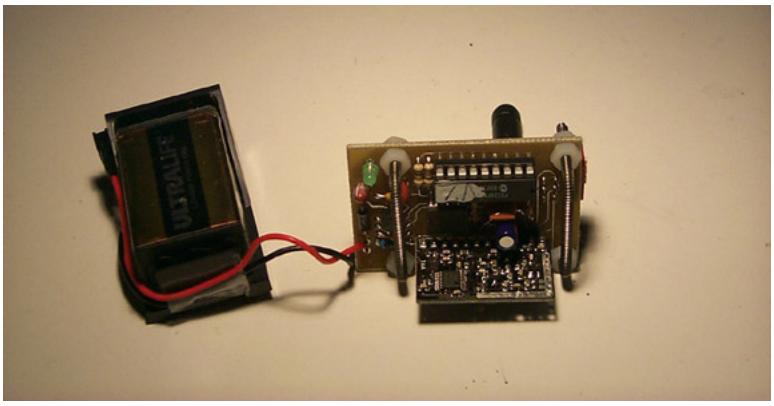


Figure 32: Voodoo Doll circuit board.

3.3.6.2 Base Design

Since the base station did not have to operate under the strict space constraints of the transmitter the design was comparatively very straightforward. We were able to "piggy-back" an iRX board with a PIC16F84 microcontroller onto the base station and use its existing antenna and power scheme.

Next we wired up the board's DB-9 serial adapter to a MAXIM 233CP RS232 level converter so that the base station could communicate through its serial port to the computer that would be hosting the embodied applications.

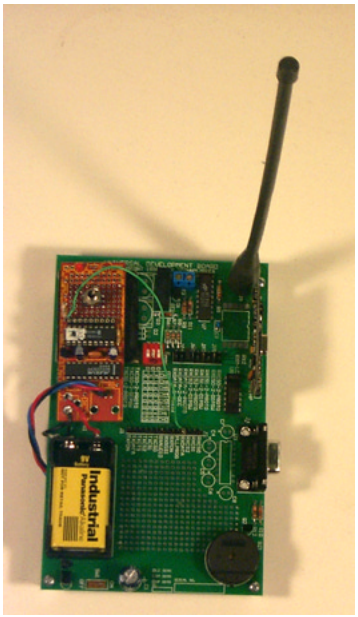


Figure 33: Voodoo Doll base station.

After some quick testing with a function generator we were able to confirm that the base station and transmitter were able to successfully communicate with one another.

Next we needed to turn our attention to the firmware design for the base station and voodoo doll's microcontrollers. The voodoo doll's microcontroller needed to decode the accelerometer's pulse width modulation stream, while the base station needed to decode the transmitter's checksum scheme and transmit information to the serial port.

After several weeks of design and debugging we arrived at a completed research prototype. It was coupled with the existing usability-feedback applications that we'd prepared for the Squeezemouse.

It had some shortcomings though. We desired to make several prototypes to give to a small pool of testers. But the cost and

complexity of the design of the Voodoo Doll were prohibitive. Consequently, we pursued cheaper alternatives.

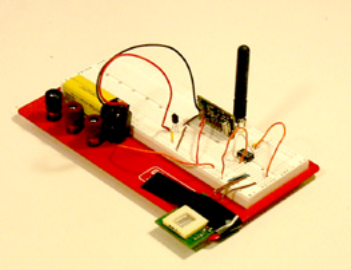


Figure 34: Analog Voodoo Doll prototype.

3.3.6.3 Analog Aggravation

One idea was to try to remove the microcontroller from the transmitter board and to directly transmit the output of an analog accelerometer to the base station. Measurement Specialties carries a 3-axis analog accelerometer. We set about building a breadboard mock up of the components that were to sit inside the voodoo doll. Initially it seemed best to try to take the three analog signals output by the accelerometer and send them into the Linx transmitter. This required the development of some glue-logic to interface the two integrated circuits. We first attempted to use a Zetex 3V voltage regulator but found that this could not source enough current for the transmitter. We then tried an elaborate scheme involving several diodes and a voltage divider borrowed from The Art of Electronics [Horowitz and Hill, 1980]. However, this scheme also did not provide enough current. We then tried to buffer the output of the accelerometer using an operational amplifier. The first op amp I selected, a National Semiconductor LM386N, also proved to require too much current from my accelerometer. Finally we settled on a Maxim MAX473, which was a nice single-source op amp that did not require much current to drive, but provided sufficient current to the RF transmitter.

After all of that trouble, we soon discovered that we had another problem altogether. The analog signals that we were transmitting were prone to interference from other RF devices transmitting on the same carrier frequency.

It turned out that ambient RF noise was very hard to differentiate from the actual shaking that my accelerometer would be recording. After speaking again to some people more knowledgeable about RF design than ourselves we learned that it is much better (as we originally had done) to transmit digital signals since checksums can be performed on them to ensure data integrity. What's more, digital signals are easier to design with and don't require the elaborate glue schemes we attempted with the analog setup.

3.3.7 Blink Detector

There is some evidence from psychophysiology that suggests that as stress increases with a task, the rate of eye blinks increases [Backs and Boucsein, 1999]. Another idea for a device that could detect stressful applications is a vision system that detects and counts eye blinks.

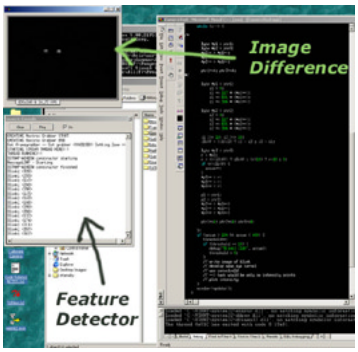


Figure 35: Blink Detector prototype.

Apparently, vision researchers already use image differencing and blink detection as a head-tracking method [Reignier, 1995]. We coded up a crude prototype and feature detector to explore the feasibility of the idea.

While this is a promising avenue for inquiry, there are some shortcomings to this approach. Users may feel uncomfortable with having cameras pointed at their face, especially if those cameras are trying to assess the user's affective state. Furthermore, although it is not real-time, the work of the CMU facial affect recognition group [Tian, 2001] has already surpassed simplistic models like just detecting eye blinks.

3.3.8 Expletive Detector

The Yelling Detector (section 3.3.3) indicated that prosodic parameters such as intensity and pitch could be used as features to detect outbursts of expletives in speech. Some preliminary work was performed to see if an expletive detector could be built.

The problem is an interesting one, which has not been well explored. Jay's Cursing in America provides an excellent introduction to the academic study of obscenity [1991].

The most straightforward approach, trying to use a speech recognizer, is problematic. Speech recognizers are often trained to recognize business speech, and consequently assign low prior probabilities to obscenities. What is needed instead is a speech system that is capable of analyzing prosody.

In an early attempt to set about creating a prosodic analyzer, I took a somewhat novel approach. In order to gather a database that could be used to train a patter recognizer, I rented several movies split into two classes: expletive-rich and expletive-less.

Figure 36: Expletive Detector voting interface.

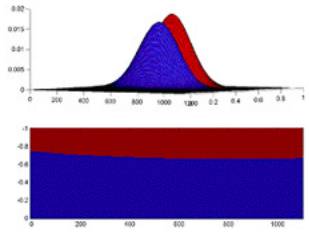


Figure 37: Expletive decision boundary.

To extract specific audio clips to use, I had a set of volunteers 'vote' on whether an utterance represented an angry, obscenity-laden, outburst.

The resulting database of sounds was used to compute the mean and variance of the sets in a two-dimensional space defined by average pitch and intensity. Using two-dimensional guassians to model the clusters, I developed the pictured decision boundary, which led greater-than-random recognition accuracy.

Afterwards energy was briefly focused on making a real-time expletive detector that could be used in experiments. Due to time constraints, I was unable to fully develop this idea, but the prototype may represent a good jumping off point for further research.

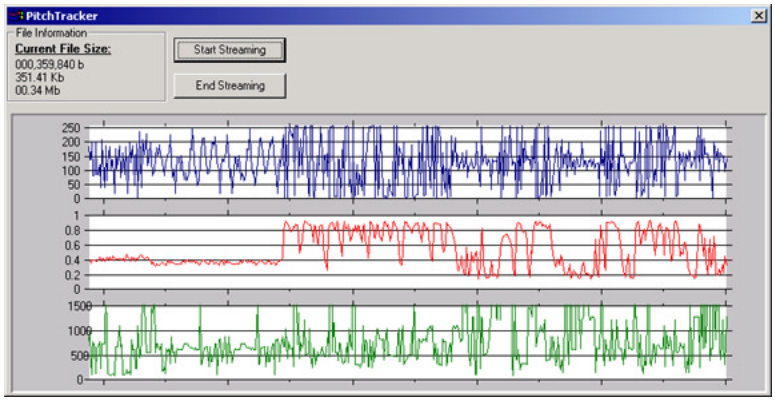


Figure 38: Real-time expletive detector prototype.

3.4 Second Revisions

In an iterative, rapid prototyping process, one revision begets another round of design and critique. In the previous section we presented several first revisions that represent shaky, tentative steps towards communicating frustration to your computer. This section deals with their revision and subsumption by more complex and more realistic prototypes.

3.4.1 Serial Swiss Army Board

After the initial hack to the iRX board (Section 3.2.2) that allowed us to collect analog data and transmit it to the computer, it became clear that we needed a custom analog data

acquisition board whose purpose was to transduce analog signals and transfer them to the serial port of a computer.

The result was a simple printed circuit board that hangs off the serial port of a PC. It uses a PIC16C711, which has an on board analog to digital converter. The firmware periodically polls the analog line and transmits converted ASCII text out on the serial port. This board was used as a conversion board for the next set of mouse designs that we evaluated. It was designed to incorporate a connector that allowed it to be used with various prototypes.

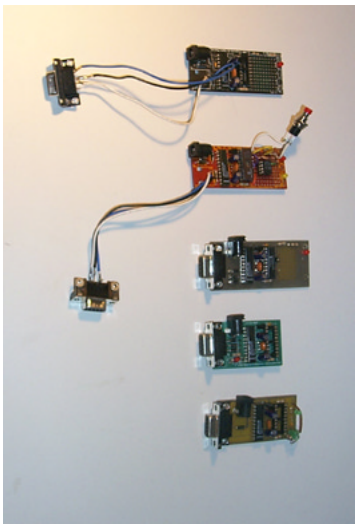


Figure 39: Serial Swiss Army board iterations.

3.4.2 Integrating Squeezemouse

Using this board, we made another revision of the squeezemouse that used a set of force sensitive resistors spread out over the surface of the mouse. These were routed into a summing junction that integrated the signal into a single signal. The resistors were covered with some neoprene foam, since foam affords squeezing. What we found is that, although the mouse had good coverage across the surface, the coverage was not uniform. Furthermore, because it was not clear where the FSRs were located, people had a lot of difficulty using it as an active interface.

3.4.3 FSR Positioning Inquiry

It soon became clear that we would need to evaluate which placements of force sensitive resistors (FSRs) were most comfortable to the user, and also did the best job of transducing squeezes. To that end a series of mice that positioned force-sensitive resistors in different promising spots were built.

Figure 40: Squeezemouse and Voodoo iterations.

Figure 41: Squeezemouse with FSR positioned beneath thumb.

After some informal experimentation, I found that if the Squeezemouse was treated as an active sensor, one of the most favorable positioning spots was beneath the thumb. It is easy to grasp the mouse and put force on the resistor in this position.

3.4.4 User Interfaces

While not traditionally thought of as a sensor, the Thumbs-Up / Thumbs-Down interface (section 3.3.4) proved to be an inexpensive and easily embeddable mechanism for users to express frustration. Consequently, we did not limit inquiry into modalities for communicating affect to tangible and physical interfaces.

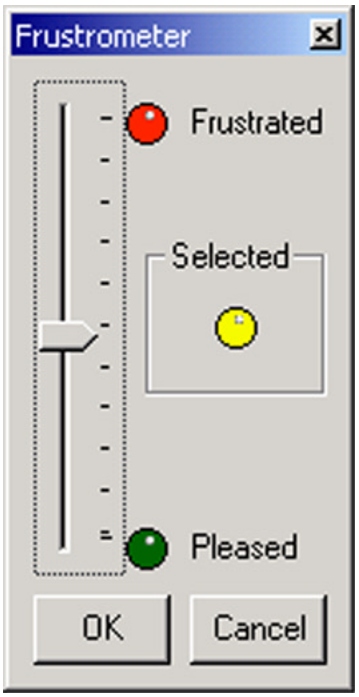


Figure 42: Early Frustometer prototype.

3.4.4.1 Frustrometer

However, Thumbs-Up/ Thumbs-Down did not provide a mechanism to communicate the severity of the usability incident, only whether a favorable or unfavorable event had occurred. Judith Ramey of the University of Washington had once mentioned that in usability tests, a cardboard "frustrometer" was used to help users express themselves [Ramey, personal communication]. Borrowing from this idea, we developed a software version. This interface allowed for a severity scale to be communicated.

3.4.4.2 Gripe

One potential application for this technology was remote evaluation of software. Consequently, we started to build "Gripe," an application embedded into the user's interface to send gripes, or feedback to usability specialists.

The first version of Gripe focused on allowing screen-shot capture to be triggered by squeezing the mouse and to be transmitted over a network. It also provided feedback about the mouse pressure input using the strip chart interface.

Gripe also acted as interface instrumentation by providing a log of which applications were running at the time of frustration incidents. More specifically, it enumerated the open dialogs, and which dialog had the user's focus at the time of a frustration incident.

3.4.4.3 Task model Revisions

The first model of gripe was something akin to sending electronic mail. But it soon became clear that there were several tasks which users might want to perform when sending feedback: typing textual comments, queuing up incidents to submit at a later time, and annotation of screenshots.

3.4.4.4 Privacy Considerations

Following discussions with several colleagues about what would make them want to use a piece offeedback software, it became clear that feedback software must allow users to maintain their privacy.

I consequently implemented tools to allow user to control which information was sent back, and to edit that information. These included a simple photo-manipulation program for erasing private content from screenshots.

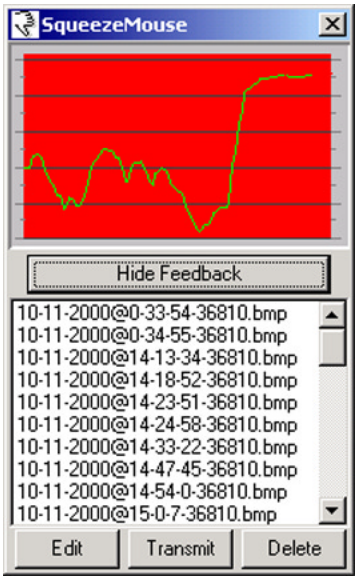


Figure 43: The first version of Gripe.

3.5 Third Revisions

3.5.1 Attentional Considerations

After informal testing, it also became clear that initial versions of gripe consumed entirely too much screen real estate. Consequently, its interface was compressed to only consume a spot on the taskbar when it was not detecting high arousal states.

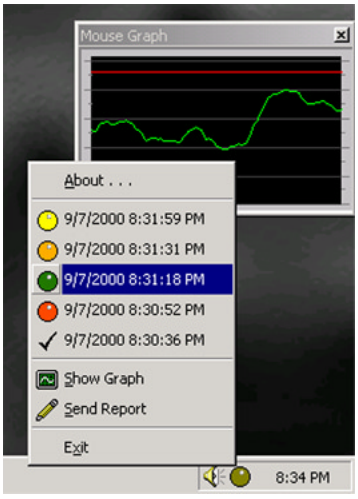


Figure 44: Task model revisions to Gripe.

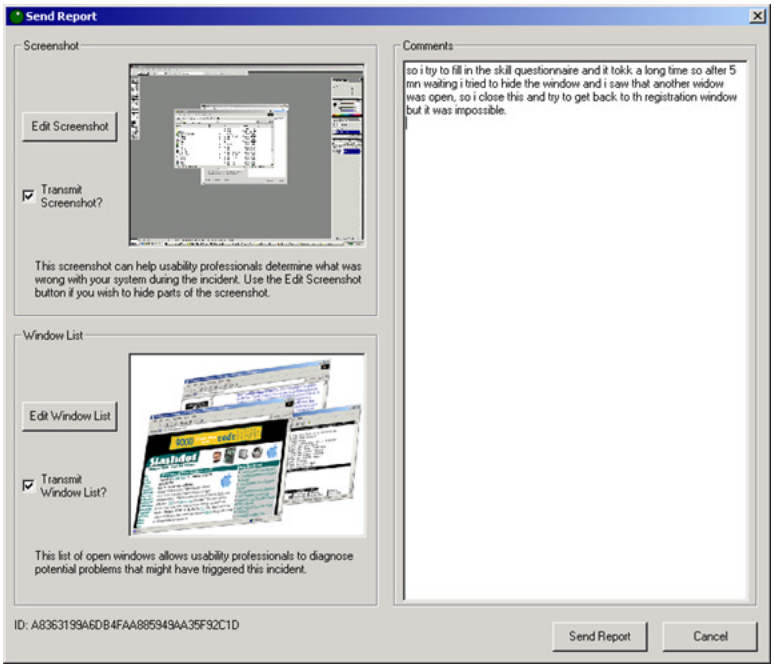


Figure 45: Privacy revisions to Gripe.

Figure 46: Unobtrusive Gripe.

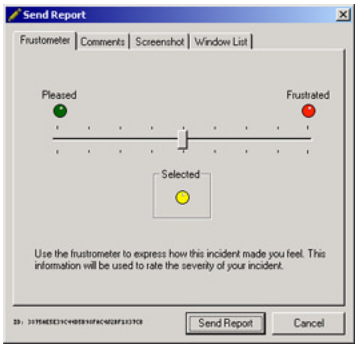


Figure 47: Frustrometer version of Gripe.

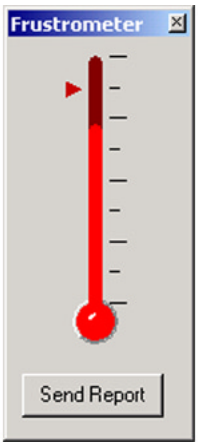


Figure 48: Thermometer squeezemouse interface.

The activated state was also compressed into a control-panel-style dialog. This was ordered to show the most pertinent information in the foreground, and the less relevant aspects in the background. Two versions of this interface were built, one that made use of the Squeezemouse, and a second that embedded the Frustrometer.

A final small revision was made to provide a much clearer metaphor when using the squeezemouse feedback software. The graphing software was replaced by a thermometer widget (not to be confused with the frustrometer). Informal testing showed that the thermometer interface was much easier to understand, since it made reference to a familiar real world object.

3.5.2 Continuous Capture

During these many iterations we began to think of the Squeezemouse less as a tangible interface and more as a sensor. Consequently, we developed a transparent Squeezemouse driver that continuously captured and labeled data collected from the mouse. Its job was to log input from the Squeezemouse to a remote network server.

3.5.3 Unobtrusive Mouse

Our earliest prototypes routed the electronics used to detect pressure along the surface of the mouse. We found the circuitry interfered with free movement of the mouse. As a result we focused on building a less obtrusive prototype that placed much of the wiring for the pressure sensors inside the mouse itself.

3.5.4 Grid-SqueezeMouse

After experimenting with mice that used just a single force sen-

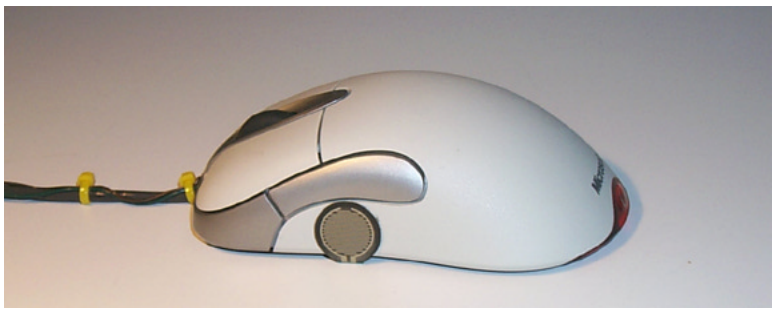


Figure 49: Unobtrusive mouse.

sitive resistor, we continued to develop the idea of treating the mouse less like an interface and more like a sensor. To this end, we made a Squeezemouse which used 144 small force sensitive resistors to cover the surface of the mouse. During the development of this mouse, we discovered that the Force Sensitive Resistors were not sensitive enough to transduce the small load placed on them by the hand resting on the mouse.

The data sheets for the Force Sensitive Resistors (FSRs) noted that components act nonlinear when they are loaded with less than 100 grams. Experimenting with a postal scale, we discovered that the FSRs operate like an open circuit with less than 30 grams of force applied.

Consequently, we abandoned development of this mouse and instead turned our attention to developing transducers that operated with less than 30 grams of force applied.

3.5.5 PressureMouse

The insensitivity of the FSRs necessitated an examination of the different materials that can be used to transduce touch. We briefly considered the use of Indium-Tin Oxide and other materials, but shied away from their use because of toxic properties or the high temperatures required to cure them.

After a hasty literature review, we learned that force sensitive resistors are composed of a conductive elastomer of some sort and electrodes. As force is applied to the elastomer or foam, it becomes more dense, and more conductive. As a result, to make more sensitive force-sensitive material, what is needed is a conductive foam that compresses under light loads.

The anti-static foam that is used to package electronic components works well as a conductive elastomer for light loads. Beginning with circular electrodes manufactured for a different variety of elastomer I constructed my own sensors.

These sensors have a greater dynamic range because the foam compresses under light loads. Unfortunately, they are not very elastic, meaning that after being loaded, they take some small amount of time to decompress.

Figure 50: Pressuremouse, equipped with eight sensors: four foam sensors at the back of the mouse and two on each side.

Figure 51: Pressuremouse tactile sensors.

These sensors, applied to several points of a mouse, allow us to determine if the user is touching the mouse or not, and how hard the user is touching the mouse. Their construction allowed for an experiment to see if user frustration can be detected passively, without requiring conscious manipulation from the user.

4 Evaluation

During the development of the various devices and interfaces in the last chapter, we began to perform more formal evaluations. These evaluations centered on performing studies containing stimuli designed to be frustrating. The first study was a comparison between different interfaces and sensors that were designed to help the user actively communicate frustration. The second pilot served to refine protocol and to examine using the Pressuremouse as a passive sensor. The final study, focused on the Pressuremouse as a passive sensor, gathered data from 16 users, and analyzed this data using signal processing and pattern recognition techniques. Finally, this chapter gives initial results on the use of pattern recognition to distinguish frustration events from non-frustration events.

4.1 First Pilot Study

Given a selection of designs for frustration sensors, we are interested in how these sensors compare. To assess the utility of these sorts of sensors, we designed an experiment to compare two different frustration sensor designs and a more traditional customer-feedback form.

We wished to compare the two sensors we developed against a baseline. After a bit of discussion we agreed that a web feedback form, like those currently in use on many websites represents one commonly used feedback mechanism. Consequently we designed a simple web form for the control group to use.

4.1.1 Methodology

Subjects for our study were solicited with flyers posted in the area around our laboratory. Spots were filled in a non-random, first-come-first serve basis, which led to the selection of nine male and four female participants. Participants in the study were read a script asking them to complete a registration sequence from Jobtrack.com:

> "We'd like you to fill out the registration sequence for a popular job search site. We'd like to strongly encourage you to use [feedback device for condition] to send feedback about any problems you have as you progress through

Figure 52: Web Form interface.

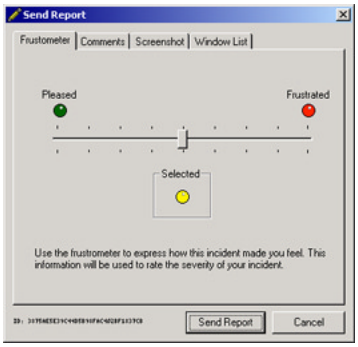


Figure 53: Frustrometer.

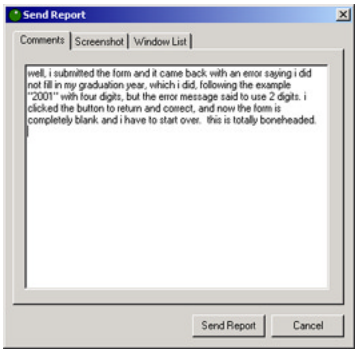


Figure 54: Squeezemouse text entry interface.

the registration process. Afterwards, we'd like to ask you to answer some questions about your experience, and fill out a questionnaire. That's basically it. We are trying to evaluate the usability of the web site. So, we need you to be a tester. Okay?"

The registration sequence consisted of six web pages asking users for information that would be typically found on a resume. The users were encouraged to evaluate the web form by providing feedback using a modality defined by the condition to which they were assigned. The conditions were:

- Web Form (Figure 52): a feedback web page consisting of a simple web form. This served as the control condition.
- Frustrometer (Figure 53): a severity slider with optional text feedback and recorded screenshots and window listings.
- Squeezemouse (Figures 54): FSR attached to a mouse along with optional text feedback and recorded screenshots and window listings.

Subjects in each condition were given a brief tutorial on how to use each of the various feedback mechanisms. Care was taken to make sure that each of the tutorials was similar both within the conditions, and between the conditions. Specifically the wording of the tutorials in the script was made to be as similar as possible, and the tutorials were set up to be approximately the same length (see Appendix B).

After the users completed the registration sequence, they were interviewed and asked to fill out a questionnaire for their condition. Each participant was only assigned to one condition, and the whole experience took less than an hour.

Unbeknownst to the participants, the web forms were designed by us to be moderately frustrating. This was achieved by violating known usability heuristics. Studies have shown that users respond poorly to varied, slow response times [Butler, 1983]. Consequently, some pages were made to load especially slowly. To further exacerbate problems, certain long forms

were designed so that no matter what sort of information was entered, the form would report errors that needed to be corrected, and forced the user to start filling in the page from scratch [Neilsen, 1999].

After being interviewed and filling out a brief questionnaire, users were debriefed and told of the deception carried out. It was emphasized that the deception was necessary, since it is very difficult to elicit emotional states like frustration if subjects know you are trying to frustrate them. Furthermore, it was emphasized that the web pages we used are not actual jobtrack.com designs, but frustrating variations designed for the purposes of our research.

4.1.2 Preliminary Results

All participants were asked on a questionnaire (see Appendix B) about the usability and responsiveness of the registration sequence. The questionnaire presented a seven-point scale from (Very Easy) to (Very Hard). For the purposes of this thesis, I've chosen to label (Very Easy) as 1 and (Very Hard) as 7. The questions and mean responses (in brackets) are shown below:

How hard was the job registration web form to use?

(Very Easy) • • (3.08) • • • • (Very Hard)

How responsive was the job registration website?

(Very Fast) • • • • (4.63) • • (Very Slow)

Since we were actually interested in the performance of our various frustration feedback sensors, the remainder of the questionnaire dealt more specifically with the sensors. For instance, the participants were asked about how difficult it was to send feedback:

"How hard was it to send feedback about the web form?"

The mean and standard deviation for each condition are summarized in Table 1.

	Mean	Std Dev
Web Form	3.0	1.7
F-Meter	1.4	0.5
S-Mouse	3.0	2.2

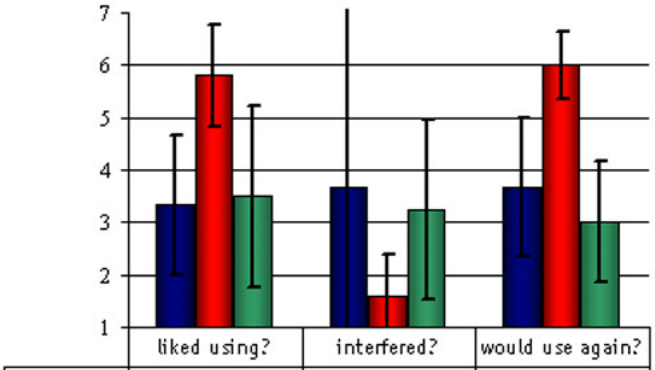
Table 1: Difficulty sending feedback.

The participants were then asked about the feedback device for their condition specifically:

- Did you like using the [Web Form, Squeezemouse, Frustrometer] feedback device?
- Did sending feedback interfere with filling out the form?

- How interested are you in using the [Web Form, Squeezemouse, Frustrometer] again?

Each condition had similar questions. For instance, the users of the web feedback form were asked "Did you like using the feedback page?" instead of "Did you like using the Squeezemouse feedback device?" The responses for each condition to these questions are summarized below:



	liked using?	interfered?	would use again?
Web Form	3.33	3.67	3.67
F-Meter	5.8	1.6	6
S-Mouse	3.5	3.25	3

Figure 55: Questionnaire data from first pilot.

Finally, in addition to these questionnaire responses the feedback provided while using the different feedback devices was examined. The number of feedback responses recorded by each device was also tallied. The mean results are summarized in Table 2

	Mean	Std Dev
Web Form	3.3	1.5
F-Meter	4.8	3.9
S-Mouse	3.5	4.4

Table 2: Number of feedback responses.

4.1.3 Analysis

What seems clear from these preliminary results is that the Frustrometer interface performed better in each of the categories we surveyed. We can see from Table 1 that the Frustrometer was reported to be not as hard to use as either the web form or the Squeezemouse. If we examine Figure 55, we see that people reported that they liked using it more than either the mouse, or the traditional web form. Furthermore, Figure 55 shows that people reported the slider's use interfered less than either the mouse or the web form. Finally, people were nearly twice as likely to indicate that they'd use the slider again.

If we perform a single tailed t-test, where the null hypothesis is that the control participants report that they are as interested in using the web form again as the Frustrometer users, we get a **p-value of 0.025**. So we should reject the null hypothesis and accept the alternative hypothesis, that the Frustrometer participants are more interested in using it again. Likewise, if we hypothesize that the Squeezemouse users are as interested in using it again as the Frustrometer users, we calculate a **p-value of 0.003**. Again, we must reject the null hypothesis and accept that Frustrometer participants are more interested in using their interface again. Of course, this assumes that our data is approximately guassian, which may not be a good assumption. Furthermore, with such low population sizes, it is ill advised to draw strong conclusions from these results. However, taken as preliminary results, they seems to indicate that the Frustrometer is a more popular interface.

So why did the Frustrometer perform better than the Squeezemouse or the web form? We theorize that it is because the Frustrometer was the most accessible, most straightforward interface, and consequently led to a lessened cognitive load when reporting feedback. In short: it was less frustrating and distracting than the other options.

The post-test interviews also provide some clues as to why the Frustrometer performed so well. One user noted that "It was convenient, easy to use." As a contrasting example, a Squeezemouse user described it as "hard to use" and mentioned that it was difficult to get the hang of squeezing hard enough. It is useful to note here that the Squeezemouse tested also included a relatively complex interface for storing and annotation feedback. The interface's complexity may have confounded results with respect to the Squeezemouse.

Overall users seemed enthusiastic about being able to send in feedback. One participant noted "I think being able to send feedback while in the middle of a process is cool and sort of prevented me from really losing my temper." Another participant enthusiastically noted, "The feedback option gave me a sense of power, in the sense that I could complain or compliment about features I dislike or like." Most users seemed to

respond positively to the convenience of accessible and easy to use feedback mechanisms: "I liked it being set up such that as soon as I realized there was a problem, I could gripe."

4.2 Second Pilot Study

After performing the data analysis for the first experiment, we became increasingly interested in seeing if the Pressuremouse would gather useful information if subjects were not told to actively manipulate it when frustrated. In short—would subjects use it differently when frustrated, without consciously thinking about the Pressuremouse? This was motivated, in part, by the realization that users did not like interrupting their work to send feedback about what made them frustrated. Perhaps, a better method for sending feedback would involve no interruption or demands on the user's attention.

Consequently, we altered the experimental design such that subjects were still exposed to frustration, but were given the newer very sensitive Pressuremouse. But they were not given any explicit training; all references to the mouse were removed from the script. The goal of the second experiment was to see if we could correctly distinguish between the data produced by the control group, who experienced no frustration stimulus, and the effect group.

Additionally, some shortcomings of the first study were remedied. The second pilot randomly assigned participants to the conditions. Demographic information was also collected about the participants, so that more concrete statements about the applicability of the data could be made.

The second pilot, however, mainly served as a mechanism for refining our test protocol and script. We were also able to adjust the parameters of our frustration stimulus. The stimulus was decided to be network delays that varied between 0 and 30 seconds. Additionally, it was decided that subjects should be placed under time pressure. Lastly, all subjects were subjected to a single loss-of-data event, which we believed would cause frustration.

During the loss-of-data event, users were told that some information on the pervious page they had entered was incorrect (see sidebar). They were then sent back to this pervious page, only to discover that all the information they had entered had been erased. The participants had no choice but to re-enter all of the data on this page again.

Video and audio recordings were captured, along with mouse pressure profiles. The continuous capture driver (discussed in section 3.5.5) was augmented with the addition of labels for whether the participants were experiencing network delay or data loss stimulus.

More specifically, participants were asked to enter a date in a four-digit format. Users who entered the date either in two or four digits were taken to page informing them they had made a mistake. When they returned to the first page to correct their mistake, users found that all of the information they inputted had been "lost." (See figure 60)

4.2.1 Anecdotal Observations

Since the second pilot study was primarily used to refine the experimental protocol, the data produced from the experiment is not suitable for rigorous data analysis. However we were able to make some observations as we ran the seven pilot participants.

The first was that our stimulus was indeed causing some response. The participants, under time pressure, felt that the site was too slow. And most felt aggravated by this:

- "I don't know if it is the connection, but it responds very slow, and I get very impatient"
- "The software is REALLY too slow" (emphasis in original)

4.3 Second Study

With our protocol more firmly established by the second pilot, we set about collecting data for our final study. The study was much less a comparative exercise between different interfaces, and more of an inquiry into the type of data created by an advanced prototype of the Pressuremouse. Superficially, there are many commonalities between this final design, and the first pilot. However the two studies had very different end goals.

4.3.1 Apparatus

The second experiment was enabled by the creation of a sensitive mouse device, which collected pressure data (see section 3.5.8). Users were placed in front of a computer with a web browser. A video camera was positioned on a tripod behind the user, and focused on the screen and their hand. Additionally, we made use of a common kitchen egg timer, to display the amount of time elapsed to participants.

4.3.2 Methodology

The design of the second full-blown experiment was a two-by-one condition, between subjects. The independent variable was the network delay applied. The dependent variable was the transduced signal captured from the squeezemouse throughout the experiment session. A total of 16 subjects were run, eight in the control condition and eight in the delay condition.

Subjects for our second study were also solicited with flyers. The flyers were posted by a service at several public kiosks at many different campuses around Boston. Participants were scheduled and randomly assigned by a web application we designed. Demographic information was collected using the same system.

Participants, (regardless of condition) were read the script below. The beginning of the script was very similar to what was used in the first pilot:

> "Thank you for coming and participating in this study! We'd like you to fill out the brief registration sequence for a popular job search site. We will be performing a usability test to see how users respond to it."

In addition, users were read a series of paragraphs to convey the time-criticalness of the task we were asking them to perform. This served to amplify the effect of the network delay:

> "After the experiment, we'd like to ask you to fill out a quick questionnaire and answer some questions about your experience. We know your time is important so we won't take too long. Okay? When you're done, the computer will notify you to get up and come get me—and

> I'll be right out here. At that point I'll give you a quick, 1-page paper questionnaire to fill out and conduct a short interview."

Here the diction ("short", "quick") was chosen to try to focus the participant's perception on time and performance. To be safe we also read the following paragraph to further reinforce the importance of completing the forms in a speedy manner:

> "During our pilot study we found that the entire experience, start to finish, should take you less than 15 minutes. Some graduate students from campus here were able to move through the webpages in 10 minutes. But we don't expect you to go that fast, we figure it shouldn't take average folks all that much longer. You should be aware of how much time you're spending. This clock will show you how much time is left."
>
> "If you run out of time, please continue until you are finished. If there's a problem, try to work through it. I'll be right outside the room. Otherwise, good luck!"

Depending on their condition, participants experienced either no delay (control condition) or randomly varying network delays (delay condition). All participants, no matter the condition, experienced a usability bug which caused them to re-enter data.

After participants completed the web site's forms, they were given a questionnaire (See appendix B). Afterwards, they were interviewed in an effort to get a more subjective assessment of their experience. The results from these, and our analysis of the data recorded from the squeezemouse are presented below.

4.3.3 Results

4.3.3.1 Questionnaire Data

As in the previous study all participants were asked on the questionnaire about the usability and responsiveness of the registration sequence. The questionnaire presented a seven-point scale for several categories (i.e. Very Fast – Very Slow).

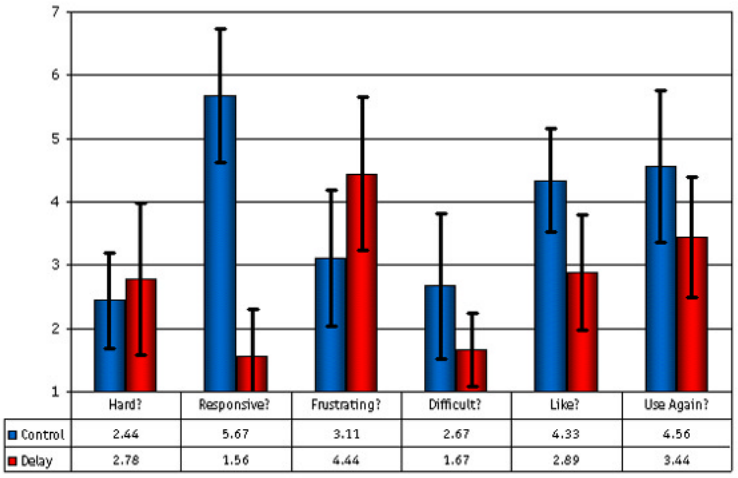
The mean results for condition A (no delays) were:
How hard was the software to use?
(Very Easy) • (2.33) • • • • • (Very Hard)
How responsive was the software?
(Very Fast) • (2.33) • • • • • (Very Slow)
How frustrating was the experience?
(Very Much) • • • • (4.89) • • (Not at All)
How mentally difficult was it to use the software?
(Very Much) • • • • (5.33) • • (Not at All)
Did you like using the software?
(Very Much) • • • • (5.33) • • (Not at All)
How interested are you in using the software again?
(Very Much) • • • • (5.33) • • (Not at All)

For condition B (0-30 second delays) the mean results were:
How hard was the software to use?
(Very Easy) • • (2.78) • • • • (Very Hard)
How responsive was the software?
(Very Fast) • • • • • (6.44) • (Very Slow)
How frustrating was the experience?
(Very Much) • • (3.56) • • • • (Not at All)
How mentally difficult was it to use the software?
(Very Much) • • • • • (6.33) • (Not at All)
Did you like using the software?
(Very Much) • • • • (5.11) • • (Not at All)
How interested are you in using the software again?
(Very Much) • • • • (4.56) • • (Not at All)

These results are summarized and shown with 95% confidence intervals in the figure 56. For the convenience of the reader, when the scale was inverted (viz. question ending in "Not at All") the graph shows the opposite.

4.3.3.2 Mouse Data

The Pressuremouse collected eight channels worth of data, and was synchronized with a label as to whether the participants were experiencing network delays or data loss. Below is an example of how Pressuremouse data log (Figure 57). The last column, which is separated by a dashed line, encodes what stimulus the user is experiencing. The data was sampled at 8 bits of resolution, at a rate of 60 Hz.



	Hard?	Responsive?	Frustrating?	Difficult?	Like?	Use Again?
Control	2.44	5.67	3.11	2.67	4.33	4.56
Delay	2.78	1.56	4.44	1.67	2.89	3.44

Figure 56: Questionnaire data from second study.

5	2	2	47	0	30	19	0	1
36	0	0	45	0	30	18	1	1
30	1	2	46	0	30	19	0	1
21	1	1	48	1	31	20	2	2
21	0	0	46	0	31	19	1	2

Figure 57: Data sampled from pressure mouse.

4.3.4 Analysis

4.3.4.1 Questionnaire Data

Looking over the summarizing graph (Figure 56), we can observe several things about the questionnaire data. First and foremost we see the effect condition, which experienced delays, found the web pages to be significantly less responsive than the control group. The findings that the software was harder to use, that they liked it less, and were less interested in using the web site again than the control were not significant at the 95% confidence level.

Both the control and effect conditions reported some frustration. This is likely to be a result of the loss-of-data event both groups experienced. However, the network-delay condition reported more frustration than the control, as would be expected (significant at p-value 0.059).

One interesting observation about the data set is that the delay condition reported that the website was less mentally difficult than the control group reported (significant at p-value 0.073).

This lends credence to the idea that cognitive load and stress are distinct from frustration.

4.3.4.2 Mouse Data

The unprocessed mouse data is 8 dimensions of 8-bit analog data captured at 60 Hz.

In order to make it possible to perform data analysis on the data set, we elected to compute a series of descriptive features

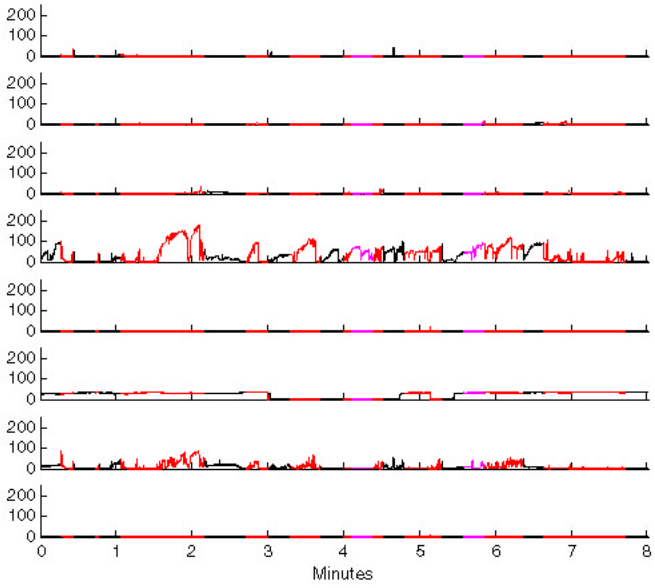


Figure 58: Sample plot of eight channels for delay condition.

on each channel. These let us more easily discern and summarize the differences in the data. I worked to develop a set of useful features that might help us discriminate between the two datasets. Below are plots combining the sum of each of the eight channels that let the reader see the visual difference between the datasets.

4.3.4.3 Pattern Recognition of Mouse Data

We broke the data set up into two separate classes of two conditions:

- Segments during loss-of-data events vs. no stimulus segments
- Segments of network-delay vs. no network delay

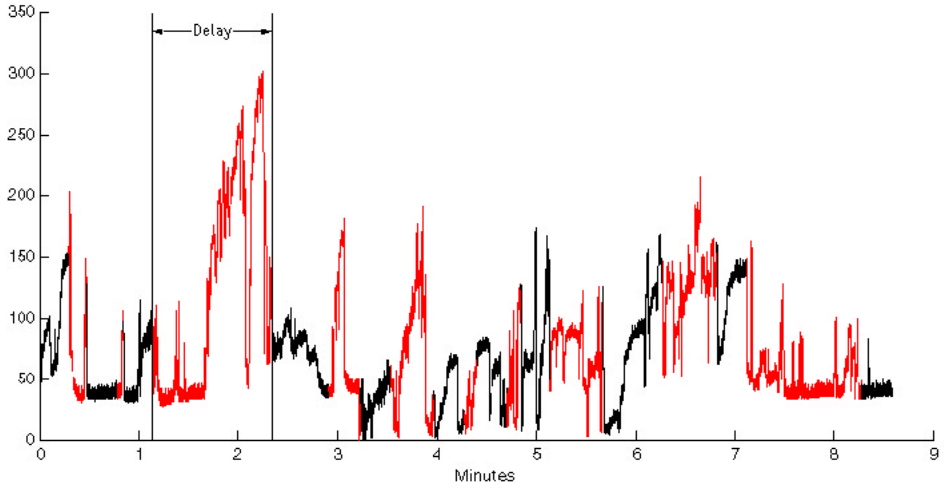


Figure 59: Sample plot of delay condition. Red denotes network delay.

We chose to screen out delay segments which were shorter than .5 seconds since they were likely not noticeable to participants. For each of the remaining sets of segments we calculated feature vectors of containing six elements:

- Zero-Crossing Rate
- Activity

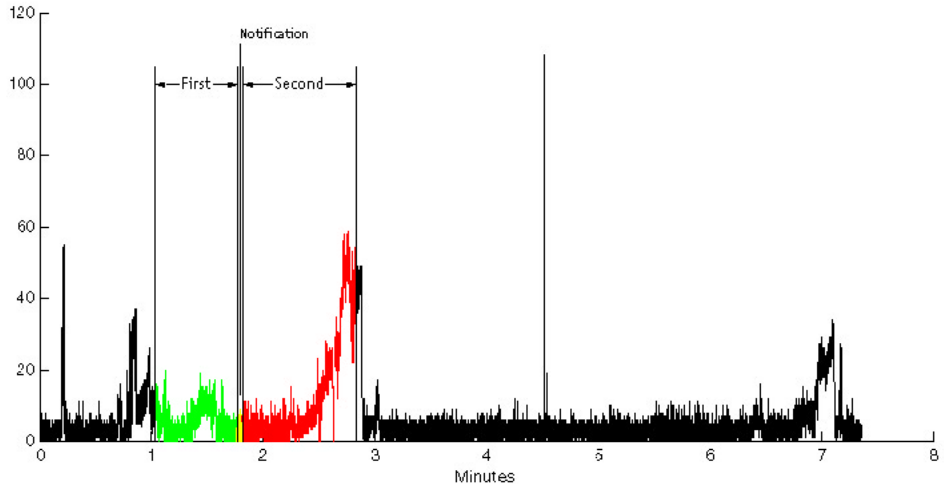


Figure 60: Sample plot of data loss from control condition. Green denotes first visit of web page. Red denotes second visit after data loss.

- Mean
- Variance
- Skewness
- Range

I will describe each feature in closer detail to aid others seeking to reproduce this work.

4.3.4.3.1 Zero-Crossing Rate

We wanted a feature to capture the high-frequency behavior of the signals. One simple approach to this would be to extract the zero-crossing rate after the time-varying mean has been removed to yield the signal:

$$y[n] = x[n] - x_{LP}[n]$$

Let $x_{lp}[n]$ be the zero-phase low-pass filtered version of x[n], where x is the raw signal for any given channel. The signal and its time-reversed version are first convolved with a low-pass filter

$$x_{LP_f}[n] = x[n] * h[n]$$

$$x_{LP_b}[n] = x[N-n] * h[n]$$

and their outputs averaged to remove the phase-delay

$$x_{LP}[n] = \frac{x_{LP_f}[n] + x_{LP_b}[N-n]}{2}$$

The low-pass filter is a simple rectangular window of length M chosen to be one second worth of data:

$$h[n] = \frac{1}{M} \quad n = 0, \ldots, M-1$$

The zero-crossing rate is then defined on the detrended signal y as follows:

$$z = \frac{1}{N} \sum_{n=0}^{N} \left| \text{sgn}(y[n]) - \text{sgn}(y[n-1]) \right|$$

4.3.4.3.2 Activity

Additionally we compute a feature that was related to how much "activity" we observed in a particular signal. This was an indicator of number of transitions between a threshold just above noise and clear pressure being applied to the mouse. We defined activity to be:

$$a = \frac{1}{N}\sum_{n=0}^{N-1}\left|\mathrm{sgn}(w[n] - \mathrm{sgn}(w[n-1])\right|$$

where:

$$w[n] = \begin{cases} x[n] - \varepsilon & if \quad x[n] > \varepsilon \\ 0 & otherwise \end{cases}$$

We empirically found a value of ε = 1 to be a good separator between activity and no activity states.

4.3.4.3.3 Mean

Another feature we selected was the mean of each of the eight channel segments. The sample mean is defined as:

$$\mu = \frac{1}{N}\sum_{n=0}^{N-1} x[n]$$

4.3.4.3.4 Unbiased Estimator of Variance

We chose to also compute the variance of the segments.

$$\sigma_u^2 = \frac{1}{N-1}\sum_{n=0}^{N-1}(x[n] - \mu[n])^2$$

4.3.4.3.5 Skewness

Since the rise times of some of the peaks in the data surrounding the events seemed to vary between the two classes, we also chose to compute the skewness s for the segments:

$$s = \frac{1}{N}\sum_{n=0}^{N-1}\left(\frac{x[n] - \mu[n]}{\sigma_b^2[n]}\right)^3$$

where:

$$\sigma_b^2 = \frac{N-1}{N}\sigma_u^2[n]$$

4.3.4.3.6 Range

Lastly, we chose to look at the range of segments :

$$r = \max(x[n]) - \min(x[n])$$

This feature reflected the peaks observed during some of the events, but also the relative flatness of other sections.

4.3.4.4 Principal Component Analysis

We reduced the dimensionality of the feature vectors compiled from the segments. Using principal component analysis, the data sets were projected from the original 40-D space down into a 5-D data space. This 5-D space captured 99% of the variation in the data sets. This aggressive reduction reflects the strong colinearity between the eight channels.

4.3.4.5 Testing / Training Division

Following this dimensionality reduction we separated the dataset into 75% training data and 25% testing data. The data was randomly separated by computing permutations.

4.3.4.6 Support Vector Machine

Next we used Scott Gunn's MATLAB implementation of Vapnik's Support Vector Machine [Vapnik, 1995]. This provided us with a decision boundary derived from our training sets with which to test recognition accuracy. The support vector machine classified the data-loss event vs. control testing data with an accuracy of 68.75%.

4.3.5 Analysis Conclusions

Following this analysis we can conclude that the stimulus caused a noticeable difference in pressure applied to the mouse. Furthermore, we can also report that a classifier trained on data collected during the second experiment was able to detect states coinciding with frustration stimulus with an accuracy rate of 68.75%. This suggests that we can detect user responses to frustration with passive sensors (and without electrodes) at a rate better than chance, but still far from ideal.

5 Applications and Future Work

5.1 Distributed Usability

One compelling application for this technology is the creation of universal and distributed usability services. If we distribute interfaces that allow people to easily send feedback about aspects of their systems, then we can begin to continuously collect usability information from them. We get a broader and more complete picture of what our users find frustrating, and one drawn from real world use settings instead of an artificial laboratory.

5.2 Interface for Reinforcement Learner

Another potential application for this work is as a front end to a machine learning process like reinforcement learning. Reinforcement learning models a process where an agent is trying to explore a space of possible actions and is rewarded by some function for positive behavior. One possible source of this reward function is, of course, a human trainer. But if this trainer is going to work with the system they must be able to easily communicate their reward feedback to the system. A command line interface where users continuously type in numerical evaluations does not seem even the least bit satisfactory.

5.3 Front End to Adaptive Generative Systems

Still another interesting application for this technology is as the front end for a generative system. A genetic algorithm, for instance, can stochastically search a space of genetic combinations. With a fitness function that evaluates a particular combination, we can gradually crawl around the space in search of progressively better combinations. This metaphor can be borrowed to allow people to creatively explore different design possibilities. We can easily imagine a set up where someone sits with a modified version of Photoshop or the Gnu Image Manipulation Program. This version takes a composition and randomly tries different filters and transformations. The user's affective response is recorded. Gradually the system can encode and learn different combinations of actions that you like. After a good amount of interaction, the system can gradually

learn to customize compositions to your previously encoded tastes. There are some similarities to relevance learning problem where we are trying to determine how relevant particular search result is to a particular user.

5.4 Usability Benchmark

Perhaps and most promising use of this technology is as an alternative usability benchmark. Time on task is one benchmark that usability specialists often use to evaluate systems. Of course, optimizing for the system that allows users to do something as quickly as possible has its problems. We may arrive at an interface that allows something to be done very quickly, but we do not know how distasteful this is for the user. For instance Time-Motion studies were often conducted around the turn of the century to try to make different assembly line tasks more efficient [Taylor, 1911]. However, these studies may have also made the worker's jobs less comfortable, since the more efficient series of actions may also be the most likely to cause repetitive stress injuries.

What is needed is a quantifiable and measurable benchmark that can be used in lieu of or to meaningfully augment subjective evaluations. If we can sense (by analyzing voice records or looking at pressure profiles) the user's continuous response to a particular piece of software, then we have the makings of a valuable design tool. It allows us to see if we are making improvements to not just the efficiency, but also the pleasantness of a particular interface.

5.5 Contextual Fusion

The sensors discussed here should not exist in a vacuum. There are a whole myriad of contextual clues about what the user is attending to, and what they intend to do. For instance, if the sensors are coupled with a gaze tracking system like Eye-R [Selker et al, 2001] assigning credit or blame to a particular interface element becomes possible. Using a Bayesian network to combine sparse information from several sensors may also allow us to come to more concrete assessments of the user's intentions.

5.6 Further Refinements

The sensor design and pattern recognition work presented earlier is in many ways a first attempt at the problem of distinguishing frustration passively. The better-than-random results are an indication that further pattern recognition work should be performed. Removing noise from the data sets and considering different features will most likely lead to higher recognition rates.

6 Summary and Conclusions

If we carefully consider the problem of making an adaptive system, we come to realize that the field has been progressing in the wrong manner. Many researchers have been building disembodied intelligences that do not take into account the subtleness or whimsy of human behavior. Without feedback and sensors to feed adaptive systems, these systems will not respond in ways that benefit users. Without any regard for the feelings of the user, these systems may result in heightened frustration and irritation for people. Clearly we need some measurable criterion around which to adapt.

We have progressed by inverting the problem of adaptive system design. We focus on the percepts rather than the intelligence in order to sense a criterion that can be used to shape behavior. We have found that frustration is a useful human behavior around which to structure change. The negative reaction frustration induces frequently coincides with the desire to alter behavior towards something more favorable.

Following surveys of psychophysiology, tangible user interfaces, and tactile sensor design, we see that many of the components exist for building sensors to detect signs of frustration many of the sensors that have been used to detect frustration force an uncomfortable relationship between the user and computer. Electrodes, for instance, may cause physical discomfort, while cameras focused on the user's face can be intrusive.

During a lengthy design process we were able to learn many things about the favorable attributes of sensors used to detect frustration. Starting with a large breadth of lo-fidelity prototypes and iterating we slowly narrowed the field of viable sensors.

The salient outcome of this process is several sensors and interfaces; but we think a more important outcome is what we learned during our evaluations:

- Participants liked having devices to communicate frustration.
- The data that was collected from both active and pas-

sive sensors can be used for redesigning and adapting systems (either by hand, or automatically).

- More specifically, there are signs of different user behavior during usability problems. With active sensors, this is clearly discernible. With passive, the picture is more ambiguous, but may potentially be clear with additional pattern recognition and context sensing.

During the first pilot evaluation of these devices we found that participants preferred the sensor designs to more traditional feedback mechanisms like web comment forms. In addition, we found that the devices and interfaces that placed the lowest burden of change on the user were the most thoroughly accepted.

The second study gave us a window into the phenomena that arise when users are intentionally frustrated. We used pattern recognition to help observe the complex ways in which people respond to frustrating stimuli. As a first inquiry into how frustration can be detected, it provided positive indications. We were able to classify data from frustration events and distinguish it from baseline data at a better-than-random rate.

A more developed form of this classifier will be a fundamental building block of user interface systems that adapt to user behavior. It will provide a measure that allows us to direct the design of interfaces towards not just more efficient, but more pleasing interactions. It could detect real emotion and quantify it in a way that can guide meaningful changes in interface design.

References

Ark, W. Dryler, C. D., Davia, J. L. (1999). The Emotion Mouse. Proceedings of HCI International Conference (Munich, Germany, August 1999).

Backs, R. W. and Boucsein, W. (Eds.). (1999). Engineering Psychophysiology. Mahwah, NJ.

Beyer, H. Holtzblatt, K. (1997). Contextual Design : A Customer-Centered Approach to Systems Designs. Morgan Kaufmann. San Francisco, CA.

Butler, T. W. (1983). Computer Response Time and User Performance, in Proceedings of CHI '83 (Dec, 1983)

Castillo, J., Hartson, H.R., and Hix, D. (1975). The User-Reported Critical Incident Method at a Glance. Virginia Polytechnic Institute TR-97-13. <http://ei.cs.vt.edu:8090/Dienst/UI/2.0/Describe/ncstrl.vatech_cs%2fTR-97-13>.

Crowder, R. M. (1998). <http://www.soton.ac.uk/~rmc1/robotics/artactile.htm>.

Ekman P, Friesen WV (1978). Facial Action Coding System, Investigator's Guide Part 2, Consulting Psychologists Press Inc.

Hartson, H. R., and Castillo, J. (1998). Remote Evaluation for Post-Deployment Usability Improvement, in Proceeding of the Working Conference on Advanced Visual Interface (L'Aquila, Italy, May 1998).

Hartson, H. R., Castillo, J., Kelso, J., Kamler, J., and Neale, W. (1995). Remote Evaluation: The Network as an Extension of the Usability Laboratory, in Proceedings of CHI '96 (Vancover, BC, April 1995) ACM Press.

Hinckley, K., Sinclair, M. (1999) Touch-Sensing Input Devices, Proceedings of Conference on Human Factors in Computing Systems (CHI '99), Pittsburgh, Pennsylvania, ACM Press, 223-230.

Horowitz, P. and Hill, W. (1989). The Art of Electronics, Cambridge, UK, Cambridge University Press, 52.

Ishii, H. and Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. Proceedings of Conference on Human Factors in Computing Systems (CHI '97), Atlanta, Georgia, ACM Press, 234-241.

Jay, Timothy (1992). Cursing in America. Philadelphia: John Benjamins.

Kirsch, D. (1997). The Sentic Mouse: Developing a tool for Measuring Emotional Valence. MIT Media Laboratory Perceptual Computing Section Technical Report No. 495.

Mueller, F and Lockerd, A. (2001). Cheese: Tracking Mouse Movement Activity on Websites, a Tool for User Modeling. Extended Abstracts of the Conference on Human Factors in Computing Systems (CHI '01), Seattle, Washington, ACM Press, 279-280.

Neilsen, J. (1999). Top Ten New Mistakes of Web Design. <http://www.useit.com/alertbox/990530.html>

Neilsen, J. (1998). Cost of User Testing a Website. <http://www.useit.com/alertbox/980503.html>.

Nicolson, E. and Fearing, R. (1993). Sensing Capabilities of Linear Elastic Cylindrical Fingers, In Proceedings, IEEE/RSJ International Conference on Intelligent Robots and Systems '93, Yokohama, Japan, pp. 178-185.

Norwood, M. R. (2000). Affective Feedback Devices for Continuous Usability Assessment. Master's Thesis, Massachusetts Institute of Technology.

OpinionLab <http://www.opinionlab.com>

Picard, R. W. (1997). Affective Computing. MIT Press.

Reeves, B. and Nass, C. (1996) The Media Equation, Center for the Study of Language and Information, Stanford University.

Reignier, P. (1995). <http://www-prima.imag.fr/ECVNet/IRS95/node13.html>

Scheirer, J., Fernandez, R., Klein, J. Picard, R. W. (2001). Frustrating the User On Purpose: A Step Toward Building an Affective Computer. To appear in special issue of Interacting With Computers.

Scheirer, J. and Picard, R. W. (2000). Affective Objects. MIT Media Laboratory Perceptual Computing Section Technical Report No. 524.

Selker, T. Lockerd, A., Martinez, J. (2001). Eye-R, a Glasses-Mounted Eye Motion Detection Interface. Extended Abstracts of the Conference on Human Factors in Computing Systems (CHI '01), Seattle, Washington, ACM Press, 179-180.

Sheridan, T. (1975). Community Dialog Technology. Proceedings of the IEEE 63, 3 (March, 1975). 463-475.

Squeekee <http://www.squeekee.com/>

Swallow, J. Hameluck, D. and Carey, T. (1997). User Interface Instrumentation for Usability Analysis: A Case Study. In Cascon '97 (Toronto, Ontario, November, 1997). <http://watserv1.uwaterloo.ca/~tcarey/casestudy.html>.

Taylor, F. W. (1911). The Principles of Scientific Management. W. W. Norton & Company, Inc., New York, NY.

Tian, Y. Kanade, T. and Cohn, J. F. (2001). Recognizing Action Units for Facial Expression Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 2, February, 2001.

UCD cam <http://miso.cs.vt.edu/~usab/remote/docs/avi98_remoteusab.pdf>

Vapnik, V. (1995). The Nature of Statistical Learning Theory, chapter 5. Springer-Verlag, New York.

WinWhatWhere Investigator. <http://www.winwhatwhere.com/index.htm>.

Appendix A (Code)

ADC conversion code

```
// File: adc.c
//
// ADC converion firmware for sensitve-mouse
//
// Carson Reynolds <carsonr@media.mit.edu>
// MIT Media Lab
// April, 2001
// Adapted from Rob Poor's iRX Hello World
#case
#include <16C74A.H>

// Configure PIC to use: HS clock, DISABLE Watchdog Timer,
// no code protection, enable Power Up Timer
//
#fuses HS,NOWDT,NOPROTECT,PUT

// Tell compiler clock is 20MHz.  This is required for delay_ms()
// and for all serial I/O (such as printf(...).  These functions
// use software delay loops, so the compiler needs to know the
// processor speed.
//
#use DELAY(clock=20000000)

// Declare that we'll manually establish the data direction of
// each I/O pin on port B.
//
#use fast_io(B)

// Standard definitions from the irx2_1 board
//
#define RS232_XMT       PIN_B1  // (output) RS232 serial transmit
#define RED_LED         PIN_B2  // (output) Red LED (low true)
#define RS232_RCV       PIN_B5  // (input) RS232 serial receive

// Macros to simplify I/O operations
//
#define RED_LED_ON      output_low(RED_LED)
#define RED_LED_OFF     output_high(RED_LED)

// Default tri-state port direction bits: all PORT B bits are
// output except for RC232_RCV (bit 5).
//
#define B_TRIS      0b00100000

// Inform printf() and friends of the desired baud rate
// and which pins to use for serial I/O.
```

```
//
#use rs232(baud=9600, xmit=RS232_XMT, rcv=RS232_RCV)

// Decleare variable to store data from one pass of ADC
//
unsigned int value;

// Counter
//
int i;

void main() {
  // since we've declared #use fast_io(B) (above), we MUST
  // include a call to set_tris_b() at startup.
  //
  set_tris_b(B_TRIS);

  RED_LED_ON;                 // reality check at startup
  delay_ms(200);
  RED_LED_OFF;

  setup_port_a(ALL_ANALOG);
  setup_adc(ADC_CLOCK_INTERNAL);
  set_adc_channel(0);

  while (1) {
      for (i=0;i<8;i++) {
        // choose sensor
        set_adc_channel(i);
        delay_ms(1);
        // get value
        value = read_adc();
        // send to serial port
        printf("%u ", value);
      }
    // delimit line of spaces
    // with newlines and loop
    printf("\n");
  }
}
```

ADC Acquisition and Labeling Driver

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;
import javax.comm.*;

public class NetSqueeze extends Thread implements Runnable,
SerialPortEventListener {
    static CommPortIdentifier portId;
    static Enumeration portList;
    static DataOutputStream os;
    static PrintWriter pw;
    static String directory;
    static String fileName = "data.txt";
    static NetSqueeze ns;
    static int label = -1;

    /*
      Matlab Color Label Key
      0 = page 1 delay
      1 = page 1
      2 = page 1 handler delay
      3 = page 1 handler
      4 = page 2 delay
      5 = page 2
      6 = page 2 handler delay
      7 = page 2 handler
      8 = page 3 delay
      9 = page 3
      10 = page 3 handler delay
      11 = page 3 handler
      12 = page 4 delay
      13 = page 4
      14 = page 4 handler delay
      15 = page 4 handler
      16 = page 5 delay
      17 = page 5
      18 = page 6 delay
```

```
*/

InputStream inputStream;
BufferedReader br;
SerialPort serialPort;
Thread readThread;
String fullLine;

static final int HUNT = 0;
static final int CHAN1 = 1;
static final int CHAN2 = 2;
static final int CHAN3 = 3;
static final int CHAN4 = 4;
static final int CHAN5 = 5;
static final int CHAN6 = 6;
static final int CHAN7 = 7;
static final int CHAN8 = 8;

int state = HUNT;

public static void main(String[] args) {
  System.err.println("-----------------------------------");
  System.err.println("|Network Squeezemouse Driver Init|");
  System.err.println("-----------------------------------");
    portList = CommPortIdentifier.getPortIdentifiers();
    while (portList.hasMoreElements()) {
        portId = (CommPortIdentifier) portList.nextElement();
        if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
            if (portId.getName().equals("COM1")) {
            // init serial port grabber
            System.out.println("* Using: " + portId.getName());
            ns = new NetSqueeze();
            break;
            }
        }
    }
}

public NetSqueeze() {
    try {
        serialPort = (SerialPort)
```

```
portId.open("SimpleReadApp", 2000);
        } catch (PortInUseException e) {
          System.out.println(e);
      }
        try {
            inputStream = serialPort.getInputStream();
          br = new BufferedReader(new InputStreamReader(inputStream));
        } catch (IOException e) {
          System.out.println(e);
      }
      try {
            serialPort.addEventListener(this);
      } catch (TooManyListenersException e) {
          System.out.println(e);
      }
        serialPort.notifyOnDataAvailable(true);
        try {
            serialPort.setSerialPortParams(9600,
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);
        } catch (UnsupportedCommOperationException e) {
          System.out.println(e);
      }

        readThread = new Thread(this);
        readThread.start();
    }

    public static void setDir(String dir) {
      synchronized (NetSqueeze.class){
          //set internal field to be used as directory
          directory = dir;

          //create new file here
          try {
            String path = directory + "/" + fileName;
            os = new DataOutputStream(new FileOutputStream(path));
            pw = new PrintWriter(os);
          } catch(IOException x)
            {x.printStackTrace();}
```

```
  }
}

public static void setLabel(int labelPassed) {
  synchronized (NetSqueeze.class){
      label = labelPassed;
  }
}

public void handleData() {
  byte[] readBuffer = new byte[20];
  // read bytes from serial port
  try {
      // 0 is not a magic number
      inputStream.read(readBuffer, 0, inputStream.available());

      // iterator over characters in string, and parse them
      StringCharacterIterator sci
        = new StringCharacterIterator(new String(readBuffer));
      for(char c = sci.first(); c != sci.DONE; c = sci.next()) {
        parse(c);
      }
  } catch (IOException e) { System.err.println(e); }
}

public void parse(char c) {
  switch(state) {
  case HUNT:
      if (c == '\n') {
        if (fullLine != null) {
            fullLine = fullLine + label;
            if (pw != null) {
              pw.println(fullLine);
            } else {
              System.out.println(fullLine);
            }
        }
        fullLine = "";
        state = CHAN1;
      }
      break;
```

```
  case CHAN1:
  case CHAN2:
  case CHAN3:
  case CHAN4:
  case CHAN5:
  case CHAN6:
  case CHAN7:
      stateMachineHelper(c, state + 1);
      break;
  case CHAN8:
      stateMachineHelper(c, HUNT);
      break;
  }
}

private void stateMachineHelper(char c, int nextState) {
  if (Character.isDigit(c)) {
      fullLine = fullLine + c;
  } else if (Character.isSpaceChar(c)) {
      fullLine = fullLine + c;
      state = nextState;
  }
}

public static void kill() {
  try {
      pw.close();
      os.close();
  } catch (IOException e) {}
  System.exit(0);
}

public void run() {
  ServerSocket serverSocket = null;
    boolean listening = true;

  try {
        serverSocket = new ServerSocket(4444);
    } catch (IOException e) {
        System.err.println("Could not listen on port: 4444.");
        System.exit(-1);
```

```
        }

      while (listening) {
          // learn to speak again.
          try {
            new SqueezeServerThread(serverSocket.accept()).start();
            yield();
          } catch (IOException e) {}
      }
      try {
          serverSocket.close();
      } catch (IOException e) {}
        try {
            Thread.sleep(20000);
        } catch (InterruptedException e) {}
    }

    public void serialEvent(SerialPortEvent event) {
        switch(event.getEventType()) {
        case SerialPortEvent.BI:
        case SerialPortEvent.OE:
        case SerialPortEvent.FE:
        case SerialPortEvent.PE:
        case SerialPortEvent.CD:
        case SerialPortEvent.CTS:
        case SerialPortEvent.DSR:
        case SerialPortEvent.RI:
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            break;
        case SerialPortEvent.DATA_AVAILABLE:
          handleData();
            break;
        }
    }
}
```

Appendix B (Evaluations):

(First Study Materials)

Briefing script

Read to all subjects upon arrival at laboratory

(comments to the administrator in bold)

NOTE: Be very friendly to the subject, but in a natural way.
Greet the subject with this script, the consent form, and a signed payment voucher, and lead them to a seat in the testing area.

If the subject asks any questions during this process, politely state:
"I'm sorry, but I'm only allowed to read from this script. I will be able to answer any questions you have when the experiment is completed."

Otherwise, read the following script:

"Thank you for coming and participating in this study!"

"We'd like you to fill out the registration sequence for a popular job search site. We'd like to strongly encourage you to use **[feedback device for condition]** to send feedback about any problems you have as you progress through the registration process. Afterwards, we'd like to ask you to answer some questions about your experience, and fill out a questionnaire. That's basically it. We are trying to evaluate the usability of the web site. So, we need you to be a tester. Okay?"

"Okay. You'll be asked to fill out the registration form for jobtrack.com. Let me show you how to use: **[depending upon condition]**

- *Customer Service Web Form*
- *Gripe*
- *Squeezemouse*

[Customer Service Web Form]
"If you have any problems, you can use the web form at http://behaved.media.mit.edu/study/comments.jsp" *[type in this URL and show user form]* It's a standard form. All you have to do is type in comments to report your frustration and hit the submit button.

[Gripe]
"If you have any problems, you can use this piece of software **[show user gripe interface]**. If you click on this icon and choose "send report" Gripe (as we call it) will record some information about what's happening. Gripe will also bring up a slider. Use that slider to report your frustration. You can also type in comments on the comments tab. **[show user comments tab]** Once you're done click the Send Report button.

[Squeezemouse]
"If you have any problems, you can use this mouse and software **[show user gripe interface]**. If you squeeze the mouse hard it will record some information about what's happening. The longer you squeeze it, the more the system registers your frustration. You can bring up an event you've captured by clicking on this icon and choosing this event. In addition you can click on the icon and choose "Send report" to start sending feedback immediately. You can also type in comments on the comments tab. **[show user comments tab]** Once you're done click the Send Report button.

Now you try. **[Watch User Try]**

When you're done, the computer will notify you to get up and come get me—and I'll be right out here—and I'll conduct a short interview give you a brief, 1-page paper questionnaire to fill out. The entire experience, start to finish, should take you less than 1 hour. Okay?

"First, we'd like to give you your payment voucher, redeemable at the cashier's office **(building 10, at the dollar bill mural in the Infinite Corridor, if they've never been). [have them fill out voucher, and have them hand it back to you to copy.]**

"Now, we'd like you to read and fill out this consent form." **[hand subject consent form, and pen if necessary. While subject fills out consent form, copy the filled-out payment voucher, and/or otherwise look busy; do not rush the subject, or make them feel nervous.] When they are done filling out the consent form, say,** "Thanks. Okay, let's get you started. Right this way."

Lead the subject into the experiment room, and offer them a seat in front of the computer. Turn on the video camera, and then say to the subject:

"If there's a problem, I'll be right outside the room. Otherwise, good luck!"

QUESTIONNAIRE A

Please answer the following questions:

NOTE: Filling out this questionnaire is very important to our research, but is entirely voluntary. Feel free to skip any question you don't want to answer.

1. How hard was the job registration web form to use?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

2. How responsive was the job registration website?

(Very Fast) ● ● ● ● ● ● ● (Very Slow)

3. How hard was it to send feedback about the web form?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

4. Did you like using the feedback page?

(Very Much) ● ● ● ● ● ● ● (Not at All)

5. Did sending feedback interfere with filling out the form?

(Very Much) ● ● ● ● ● ● ● (Not at All)

6. How interested are you in using the feedback page again?

(Very Much) ● ● ● ● ● ● ● (Not at All)

7. Prior to today, have you ever been a participant in a usability test?

● Yes ● No

8. What are your impressions of the whole experience of filling out the form, and sending in feedback?

QUESTIONNAIRE B

Please answer the following questions:

NOTE: Filling out this questionnaire is very important to our research, but is entirely voluntary. Feel free to skip any question you don't want to answer.

1. How hard was the job registration web form to use?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

2. How responsive was the job registration website?

(Very Fast) ● ● ● ● ● ● ● (Very Slow)

3. How hard was it to send feedback about the web form?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

4. Did you like using the Gripe feedback software?

(Very Much) ● ● ● ● ● ● ● (Not at All)

5. Did sending feedback interfere with filling out the form?

(Very Much) ● ● ● ● ● ● ● (Not at All)

6. How interested are you in using Gripe again?

(Very Much) ● ● ● ● ● ● ● (Not at All)

7. Prior to today, have you ever been a participant in a usability test?

● Yes ● No

8. What are your impressions of the whole experience of filling out the form, and sending in feedback?

QUESTIONNAIRE C

Please answer the following questions:

NOTE: Filling out this questionnaire is very important to our research, but is entirely voluntary. Feel free to skip any question you don't want to answer.

1. How hard was the job registration web form to use?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

2. How responsive was the job registration website?

(Very Fast) ● ● ● ● ● ● ● (Very Slow)

3. How hard was it to send feedback about the web form?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

4. Did you like using the Squeezemouse feedback device?

(Very Much) ● ● ● ● ● ● ● (Not at All)

5. Did sending feedback interfere with filling out the form?

(Very Much) ● ● ● ● ● ● ● (Not at All)

6. How interested are you in using the Squeezemouse again?

(Very Much) ● ● ● ● ● ● ● (Not at All)

7. Prior to today, have you ever been a participant in a usability test?

● Yes ● No

8. What are your impressions of the whole experience of filling out the form, and sending in feedback?

(Second Study Materials)

Briefing script

Read to all subjects upon arrival at laboratory

(comments to the administrator in bold)

NOTE: Be very friendly to the subject, but in a natural way.
Greet the subject with this script, the consent form, and a signed payment voucher, and lead them to a seat in the testing area.

Apparatus:

- **Video camera focused on user's screen over shoulder**
- **Pressure Sensitive Mouse**
- **Instrumented software which collects mouse data**
- **Timer Clock**

If the subject asks any questions during this process, politely state:
"I'm sorry, but I'm only allowed to read from this script. I will be able to answer any questions you have when the experiment is completed."

Otherwise, read the following script:
"Thank you for coming and participating in this study! We'd like you to fill out the brief registration sequence for a popular job search site. We will be performing a usability test to see how users respond to it."

"With your consent as you use the job search site audio, video, and mouse input will be recorded."

[Hand subject consent form, and pen if necessary. While subject fills out consent form, Look busy; do not make them feel nervous.] When they are done filling out the consent form, say,

"Next, we'd like you to sign your payment voucher. Once we make a copy for our records, We'll give you the original which is redeemable at the cashier's office immediately following the experiment. **(building 10, at the dollar bill mural in the Infinite Corridor, if they've never been)**

[have them sign voucher, and make copy for our records.]

[setup mouse, and ensure all the camera is on]

"After the experiment, we'd like to ask you to fill out a quick questionnaire and answer some questions about your experience. We know your time is important so we won't take too long. Okay?"

"When you're done, the computer will notify you to get up and come get me—and I'll be right out here. At that point I'll give you a quick, 1-page paper questionnaire to fill out and conduct a short interview."

[Bring up http://arsenal.media.mit.edu/study/start.html, and select their condition]

[Be a little insulting here]

"During our pilot study we found that the entire experience, start to finish, should take you less than 15 minutes. Some graduate students from campus here were able to move through the webpages in 10 minutes. But we don't expect you to go that fast, we figure it shouldn't take average folks all that much longer. You should be aware of how much time you're spending. This clock will show you how much time is left."

[Start the clock at 15 mins (go to 30 and then back)]

"If you run out of time, please continue until you are finished. If there's a problem, try to work through it. I'll be right outside the room. Otherwise, good luck!"

QUESTIONNAIRE

Please answer the following questions:

NOTE: Filling out this questionnaire is very important to our research, but is entirely voluntary. Feel free to skip any question you don't want to answer.

1. How hard was the software to use?

(Very Easy) ● ● ● ● ● ● ● (Very Hard)

2. How responsive was the software?

(Very Fast) ● ● ● ● ● ● ● (Very Slow)

3. How frustrating was the experience?

(Very Much) ● ● ● ● ● ● ● (Not At All)

4. How mentally difficult was it to use the software?

(Very Much) ● ● ● ● ● ● ● (Not at All)

5. Did you like using the software?

(Very Much) ● ● ● ● ● ● ● (Not at All)

6. How interested are you in using the software again?

(Very Much) ● ● ● ● ● ● ● (Not at All)

7. Prior to today, have you ever been a participant in a usability test?

● Yes ● No

8. What are your impressions of the whole experience of using the software?


Earn $15 in an hour
. . . for using the web
Participate in a study
at the MIT Media Lab,
receive $15 for about
an hour's worth of time.
A modest amount of computer
and surfing experience needed
Help us evaluate a website
Sign Up: http://arsenal.media.mit.edu