CS376 Project Milestone 1
Lightweight Application Instrumentation for User-Initiated Feedback in Off-the-Desktop Interfaces

1. ABSTRACT

[A ~100-150 word summary of the interface you plan to prototype, the hypothesis of the value that it provides the user, and how you plan to evaluate if your hypothesis is true or false.]

Effective user-centered application design requires knowledge about user needs and real-world system usage, among other factors. Existing research has mostly focused on observational or predictive short-term evaluation of screen-and-mouse/keyboard-based applications. The proposed project will investigate ways to gather application usage data in an entirely different context: situations where the user will be actively involved in providing feedback (possibly infrequently) over an extended period of time during her real work with novel, off-the-desktop applications. Motivation for this approach (beyond looking for low-hanging fruit) comes from the following observations: users are good judges of when to report interesting incidents or software failures and may be willing to report these incidents if associated cost is low; long-term evaluation in the "real world" can provide deeper insight into usage than short-term studies; novel interfaces that lack standardization have a greater need for user evaluation. Goals for the project are to devise an abstract framework of how to collect and report user data in such a context; to possibly develop a software toolkit encoding the instrumentation principles in software; to design one instrumented off-the-desktop user interface; and finally to study efficacy of the proposed technique(s) through a small-scale comparative evaluation.

2. TASK ANALYSIS

[Who are the users? What tasks will the users need to perform? What new tasks do they desire to perform? Where are the tasks performed? How are the tasks learned? What set of tools does the user have now? How often do your users perform the tasks? What happens when things go wrong? What's the relationship between the user and his or her data? How do your users communicate with each other? With whom do they communicate? Use the questions above to guide your task analysis. This should be one or two paragraphs..

There are two target audiences: developers that create interfaces and the users that then interact with these interfaces. Let us deal with each in turn.

Developers or interface designers need to understand their target audiences. One way to achieve this goal is through direct data collection from real world users with instrumented applications. For effective instrumentation, developers need theoretical guidelines and practical tools to help them add the functionality to their existing hardware or software projects. A published framework for how to think about long-term lightweight user data can be employed to guide design decisions by hardware and software engineers. Software developers can also benefit from an API that exposes functionality to facilitate the collection and reporting of user information back to the software company. An abstract API would be useful to have developers follow a specific process for recording and reporting data while a concrete API for a given platform could focus on the specific interaction techniques afforded by that platform.

Users in this project would be long-term beta testers or early adopters of novel computing interfaces. They will perform real tasks that require frequent interaction with the chosen hardware device. The added functionality this project provides is a mechanism to record meta-information about the user's interaction. The recording of the information is user-initiated to let the user maintain (a sense of) control over data collection. Easy, quickly accessible commands allow the user to give short feedback without having to break out of interaction with the device. Feedback functionality is always accessible – no matter if the device is connected to a network or not. Data will be sent back (after user-review to give consent over what exactly will be transmitted) to a support provider or developer which would in turn send a response back to the device or user after some kind of analysis. Alternatively, the data can be used by the application internally to adapt to user habits.

3. IDEATION

[Sketches and Storyboards of your ideas. This will be the main part of your grade for this assignment. We want to see lots and lots of ideas. Brainstorm! Not all your ideas have to be related to your proposed system. You are not required to digitize sketches. Sketches should be in whatever format enables you to work quickly; we recommend pen and paper. They should be accompanied by short blurbs to explain each idea. We will be evaluating sketches based on two criteria: volume and breadth. Please also turn in your half-baked and "dumb" ideas. The goal of this section is to encourage brainstorming and ideation. Come up with lots of ideas, and have them span a broad design space. Some will be more compelling than others, and that's okay.]

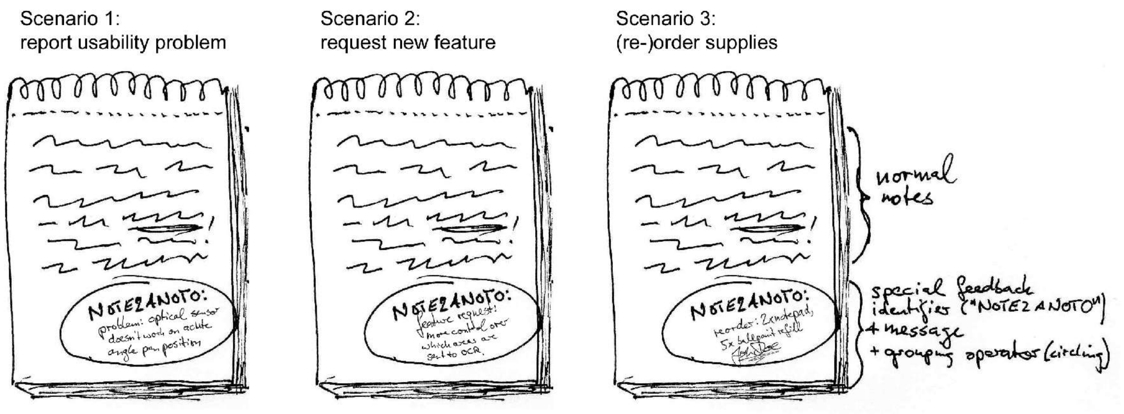
Ideas for possible implementation in specific interfaces:

3.1 "Note2Anoto" functionality for a digital pen:

The Anoto digital pen, commercialized by Logitech under the name "io," uses an optical sensor embedded in a standard ball-point pen to track user writing on specially watermarked paper. In addition to the normal physical record of ink on paper, the Anoto pen can transfer all stroke information as images to a computer using USB or Bluetooth interfaces. OCR may then be used to turn the handwritten notes into editable text. I suggest to introduce a special feedback identifier – a unique symbol or the string "NOTE2ANOTO" that users can employ anywhere in their notebooks to compose messages to be sent back to Anoto/Logitech. Upon synchronization with a PC, the feedback information will be extracted and presented for verification before being emailed back to the company. Three types of feedback come to my mind:

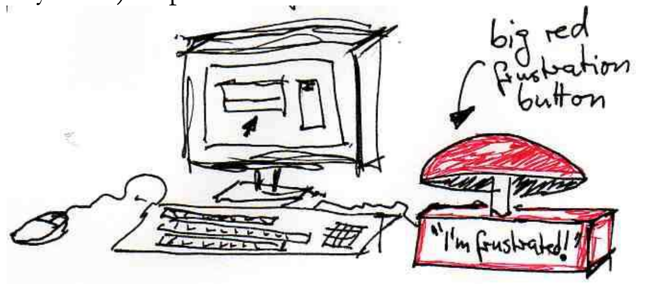
1. Usability problems: Hardware problems such as optical recognition errors if the pen is angled too much or when a faint dot pattern cannot be discerned in bright outdoor light. Software problems like jumbled OCR recognition if text was written down in changing orientations on the page.
2. Requests for feature extensions: Users can submit features they would like to be added in the future. These ideas may only come up after some time as usage patterns change with mounting user expertise. Example: Users may want to have a feature in the recognition software to select certain regions of a page to be sent to OCR while others (diagrams) should be kept as images.
3. Sales: consumables such as notepads and pen refills can be ordered directly without having to go through a shopping website. Economics may be an important argument for companies to add features which acknowledge to the user that their product has imperfections.

By providing the functionality directly on the paper notepad, user work flow is disrupted only minimally – the feedback note can be scribbled anywhere at any point of the normal writing task. I have personally ordered one of these pens, so the hardware can be provided.


Scenario 1:
report usability problem
Scenario 2:
request new feature
Scenario 3:
(re-)order supplies
NOTE2ANOTO:
problem: optical sensor doesn't work on acute angle pen position
NOTE2ANOTO:
feature request: more control over which areas are sent to OCR.
NOTE2ANOTO:
reorder: 2x notepad, 5x ballpoint refill
normal notes
special feedback identifier ("NOTE2ANOTO")
+ message
+ grouping operator (circling)

3.2 Capturing user frustration with dedicated hardware:

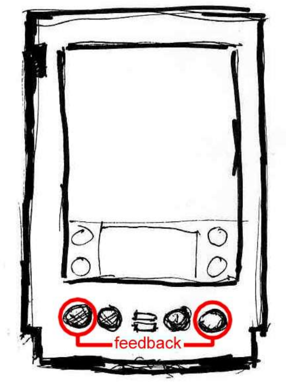
Not an off-the-desktop idea: Add a dedicated button that users can hit (hard) when the software they are using is frustrating them (cheap-o alternative: reprogram and paint one of the additional keys on a multimedia keyboard). This voluntary feedback can be synchronized and combined with internal application state data to help developers infer what exactly the problem was. The lightest implementation would send the data without any additional user information; a more involved prototype could pop up a simple text-entry area where the user can describe the encountered problem in a few words ("I can't find the #$%* command to center my text") to provide additional context.



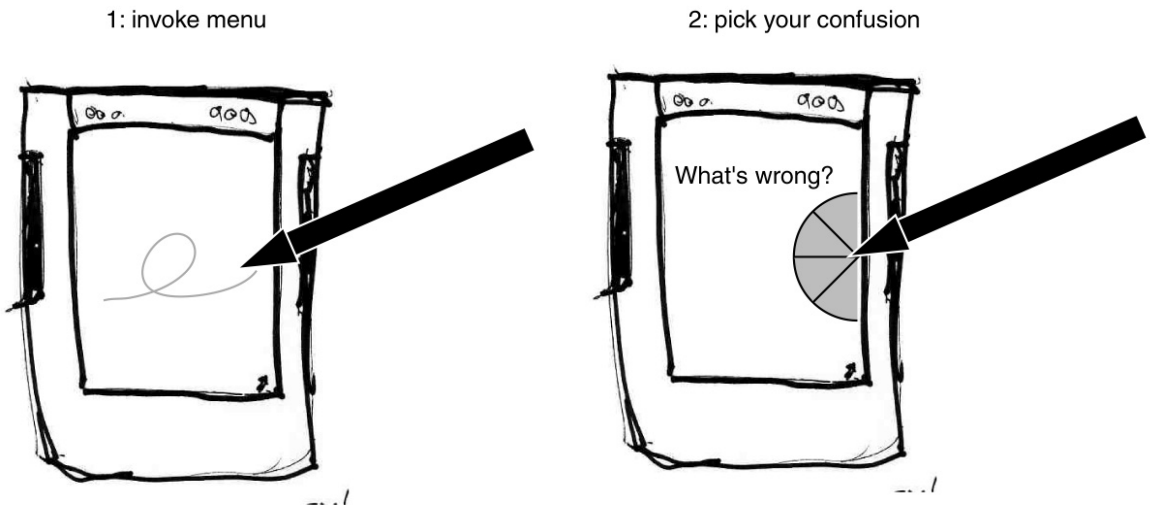
I am partial to the idea of adding explicit hardware to indicate confusion/system problems in general. Let designers acknowledge the imperfection of all hard- and software and accordingly create "failure-aware designs" - devices that come equipped with easy methods for users to record their gripes.

3.3 Capturing user confusion on Tablet PCs:

We can extend the previous example by moving to a tablet-based interface or a PDA.. Since real-estate for physical buttons is expensive here, we could assign a combination of existing buttons – an analogy to the "ctrl-alt-delete" on keyboards – to invoke the feedback function. To make users aware of this function, the relevant buttons would be visually grouped, for example by printed connected red outlines. Additionally, a slightly more complex user-reporting scheme can be adopted: a taxonomy of different kinds of user confusion could be developed (note the difference between "I don't understand the application" vs. "The application doesn't understand me") and dedicated reporting facilities for each kind of confusion would be added to the device.



In an alternative implementation on a Tablet PC, we can invoke a marking menu through a specific pen gesture or holding the pen in place in a particular screen region for a few seconds. The marking menu would once again display the confusion choices and data can again be correlated with application state variables behind the scenes.



3.4 The “Red card” for vision-based recognition interfaces:

Camera-based gesture tracking is error-prone and often recognition mismatches are mysterious to users. The idea here is again to keep the user interacting with the device while providing feedback. The recognition setup could come equipped with a bright red card which users can hold up in front of the camera to give the recognizer a time-out and start feedback mode. The camera (with microphone) would then record the user speaking and gesturing, explaining her problem. Example: “I just moved like this (waves left arm up and down) and last time I did it the application responded with X, but now it responded with Y.” The video can then be sent to the computer vision algorithm designers.

3.5 Multimedia complaints from digital camera users:

An increasing number of digital cameras feature small microphones to annotate pictures. When facing a problem with the interface, photographers could press a “gripe” button (or button combination) to voice-record a description of their problem. If the problem is related to the image last taken, the image could be flagged together with the voice recording. During the next transfer of pictures from camera to host computer, the specially flagged images/audio recordings could be forwarded to the camera manufacturer. There are obvious problems with sending your private pictures on, but I don't think users are willing to reproduce a problem by taking a second picture of some still life.

3.6 Reinforcing the habitual mind – but doing so selectively:

Location-aware mobile devices such as smart cell phones or GPS-equipped PDAs can use georeferencing to learn geographical and temporal user patterns – for example, Bjoern checks the Caltrain schedule on Fridays shortly before heading to the Palo Alto train station. Anna tends to start the calendar application on her SideKick and scrolls to the next week before calling her co-worker on Thursdays. The devices can learn these associations but shouldn't do so indiscriminately – some patterns may be accidental, others may be conscious but unwanted (automatically going to nytimes.com to check the news when opening up a web browser). If the user is given the choice to control when something is learned (“pay attention now and remind me next time” button) and when something is ignored, a closer match between user wants and application behavior can be found.

Ideas not yet explored sufficiently:

- Using the remote control of Windows Media PCs as an input device (the TIVO thumbs up/ thumbs down buttons already provide similar functionality)
- How can we integrate the reported data back into the development process (think SUEDE)?

EVIDENCE

[Present some evidence that your idea is a good one. What observations did you make during your contextual inquiries that support your idea? What other systems exist that support similar tasks?]

Some evidence for the usefulness of the proposed approach was derived from a cursory review of literature. Automatic usage data gathering and remote evaluation of deployed software is not a novel idea – first systems were implemented in the 1980s [1]. Hilbert and Redmiles [2] present a detailed taxonomy and an exhaustive survey of GUI-based usability

information gathering. Castillo et al. [3] conducted a study showing that users were capable and willing to report their own critical interaction incidents with software. The same research group also wrote about the potential usefulness of conducting ongoing studies of deployed software [4]. Newman et al. [5] note in their paper about an informal pen-based interface that it was difficult to evaluate the origin of a specific usability problem (pg. 312) – additional, more detailed problem reporting tools within the software may have helped.
The shift of focus from GUI-based applications to novel forms of interaction resulted from informal discussions with Scott Klemmer. Discussion of ideas with other CS graduate students not involved in HCI research were a good indicator if these ideas had any appeal to end users.

FURTHER EVIDENCE

[How will you collect further evidence to support your intuition that your system is a good one?]
I will conduct a more thorough literature review and try to get as much feedback from the instructor and fellow graduate students. Since the suggested techniques are all bound to a specific device, rapid testing techniques in different media such as paper prototyping do not appear promising.

EVALUATION PLAN

[How will you evaluate your hypotheses? What type of user tests? How do you plan to set up your user tests? How many people will participate in your study?]
If my Anoto pen is delivered next week as planned, there should be enough time to develop a working image segmentation-based prototype for the Note2Anoto idea. Alternatively, the computer vision feedback system could be implemented – I have the hardware – but I would need to tap into existing recognition code as well as have a access to an application that uses gesture recognition. To evaluate the usefulness of lightweight feedback, I would have a small number of users (think CS147 students) split into two groups – one could send usability feedback with the pen or per video, the other group would be instructed to submit problems via a submission form on a website. The hypothesis is that lightweight feedback on the device itself will be used more frequently. The kind of long-term interaction envisioned in the initial design idea will be impossible to carry out before the end of the quarter.

REFERENCES

[1] Dan R. Olsen and Bradley W. Halversen. Interface usage measurements in a user interface management system. In Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software, pages 102–108. ACM Press, 1988.

[2] David M. Hilbert and David F. Redmiles. Extracting usability information from user interface events. ACM Comput. Surv., 32(4):384–421, 2000.

[3] José C. Castillo, H. Rex Hartson, Deborah Hix. Remote usability evaluation: can users report their own critical incidents? CHI 98 conference summary on Human factors in computing systems, pages 253 - 254 . ACM Press, 1998

[4] H. Rex Hartson, José C. Castillo, John Kelso, Wayne C. Neale. Remote evaluation: the network as an extension of the usability laboratory. Pages: 228 – 235. ACM Press, 1996

[5] DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice, Mark W. Newman, James Lin, Jason I. Hong, James A. Landay, Human-Computer Interaction, 2003. 18(3): pp. 259-324