Usability Evaluation with Minimal Observation Impact

Stephen Bell, Alisdair McDiarmid, James Irvine, University of Strathclyde; Nigel Jeffries, Vodafone

***Abstract*—One of the challenges for UI evaluation of portable devices is recording user interactions for later study. Common approaches include laboratory monitoring, user-wearable video recording equipment, and human monitoring assistants. However, all of these recording approaches will affect the user's interactions, and therefore reduce the validity of the results. We propose that handheld devices are becoming powerful enough to observe the user as well as be the object of the study, given certain technical innovations described below. This paper describes an approach to recording user interaction sessions for use in usability evaluation studies on handheld devices.**

***Index Terms*—Graphical user interfaces, Software, Recording, Video recording.**

I. Background

Usability testing is an important aspect of product research and development. It can be used to verify a proposed new interaction method, or to test the quality of a near-market device. For handheld device testing, we categorise these studies into three groups: laboratory testing, user-recorded interactions, and third-party monitoring.

Laboratory testing is the traditional approach to usability studies. A user is introduced to a controlled environment, given a device to use, and asked to perform some tasks. Video cameras can be used to study exactly how the user interacts with the device during the test. Stoica et al. [1] propose that while this gives excellent data, the context and the surroundings as well as the other people around play an important enough role to merit field-testing.

Field-testing of devices can give a more realistic insight into their practical performance. Commonly, field usability tests require the test subject to wear monitoring equipment to record how they interact with the device under test [2][3]. While the device is being used in a realistic environment, the presence of the monitoring equipment may modify the user's behaviour significantly. It is difficult to act naturally while wearing a helmet-mounted camera, especially when walking down a busy street.

An alternative field testing approach uses a third-party to record the user's behavior [4]. While this is less of a physical burden on the test subject, it still affects the user's normal environment. It also doesn't scale to large numbers of test subjects, and is an expensive option.

II. Handheld Device Monitoring

The handheld devices that are being tested for usability are becoming more powerful, with ever-increasing CPU, RAM, and mass storage capacity. We believe that it is now possible to place the burden of recording a user's device interactions on the device itself. This paper describes our motivation, initial approach, and future direction for this work.

III. Example Application: Instant Knowledge

Many employees carry powerful mobile handheld devices with them wherever they go. These devices are usually provided by the employer, and are to be used for work purposes. Instant Knowledge (IK) leverages the power of these devices for capturing the informal interactions between colleagues [5]. The interactions recorded by a user's device can be used to build a map of their contacts within the organisation and estimate the strength of their relationships with their colleagues [6]. This information can then be used to connect employees with other, similar people through friend-of-a-friend style introductions, increasing the efficiency of collaboration.

A near-market IK product will require usability testing to study its performance in real-world situations. Due to IK being intended for mobile devices, interaction sessions will often be sparsely distributed throughout the day, but have high interaction frequency. These sessions could span multiple contexts of use, and the user-device interactions could be influenced by many environmental factors such as temperature, ambient noise, and device acceleration. This context information is central to several aspects of the IK system, and it is therefore important to be able to perform usability testing in the field.

IV. Method

To capture a user interaction event, we record three data: the screen state preceding the interaction, the interaction itself, and the screen state after the interaction. In doing so, we are able to later reconstruct a replay of the user interaction, showing the trigger for the interaction, displaying the user's input, and then the result of this input on the screen.

To record the screen state preceding the user interaction event, a ring buffer approach is used. The ring buffer is continuously updated with the state of the screen, with the oldest screen state being discarded after it leaves the fixed-size buffer.

When a user interaction event is detected and recorded, the contents of the buffer are moved to mass storage. The ring buffer is then allowed to refill completely, at which point it is again stored to disk. This results in the recording of the events leading up to a user interaction, the interaction event itself, and the moments after the user interaction occurs (Figure 1).

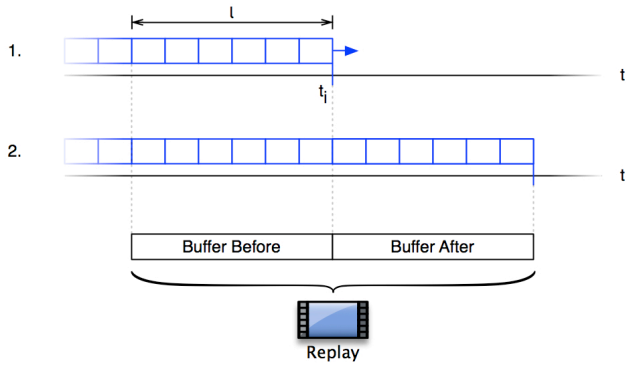

Figure 1: Recording l screen states before and after an interaction at time t_i.

Interaction events will often occur in quick succession. If a second user interaction event occurs before the ring buffer has refilled completely, the partially full buffer is stored. The ring buffer is then allowed to refill as normal, unless interrupted again. In this way, a normal sequence of user inputs will result in a reconstructed continuous replay of pre-conditions, inputs, and results, without any overlaps.

V. Prototype Implementation and Evaluation

The technique described here is currently in prototype implementation. We are using a Nokia N810 Internet Tablet as the demonstrator device for the IK project, and as a mostly open-source Linux device it is well-suited to development of these ideas.

A current prototype uses custom software to record and stream the device frame buffer into a ring buffer, in compressed format. User input data is captured by monitoring X11 events, including some N810-specific extensions. These data and associated metadata are recorded to a micro-SD card attached to the device, and a replay application on a desktop computer reconstructs the user experience.

Nielsen [7] suggests that 10 seconds is about the limit for keeping the user's attention. GNOME uses this in the Human Interface Guidelines for their Desktop Window Manager [8]. The recording application uses this metric for the amount of time stored before and after an interaction. However, the time is variable and may change after testing of the implementation has been conducted.

A. Recording User Input

User input is recorded by hooking into X11, intercepting events before they are passed to their respective processes. Each of the detected events has a variety of associated data, including the process ID of the receiver, and the key code of the pressed key (which can be converted into an ASCII character) or the co-ordinates of the touchscreen event. Both presses and releases are detected, so it is possible to determine the duration of a user interaction. The interactions are stored in a database for later reconstruction of the session. Fields include a high-resolution time stamp, the type of interaction, and type-specific data for the interaction.

B. Recording Screen State

The screen state of the device is recorded by reading directly from the frame-buffer. The frame-buffer can be copied to RAM at just below 30 frames-per-second, which is consistent with the refresh rate of the display. However, the limitation on the N810 is on the speed of the disk, not the RAM.

The time taken to write a raw frame to disk was measured at 0.27 seconds, giving a measurement of 3.7fps while the device is under no load. Performance is worse when other applications are using CPU or disk time.

The ring buffer writes frames to disk when they are to be overwritten. Writing out all of the frames in the buffer immediately after a user interaction event has occurred would cause a CPU usage spike every time a user interaction occurs. This would require asynchronous reading from and writing to the buffer, and therefore a flexible buffer size. Writing a single frame to disk just before it is freed maintains a fixed buffer size, and more consistent CPU and disk utilisation for the recording process.

In order to counter the slow disk write speed, the raw frames are compressed on-the-fly before being inserted into the ring buffer. Two factors affect the frame-rate: the execution time and the compression ratio of the algorithm. We wanted to select the best possible compression algorithm to maximise the write performance of the system. This is a trade-off between CPU time used and compression level achieved.

A variety of compression algorithms were tested: fastlz [9], gzip [10], liblzf [11], and PNG. Twenty unique images for three commonly used applications were used: the desktop, the email client, and the chess application. Sample images are shown in Figures 2, 3, and 4. The average compression ratio and execution time for each algorithm on each set are shown in Table 1 and Table 2.

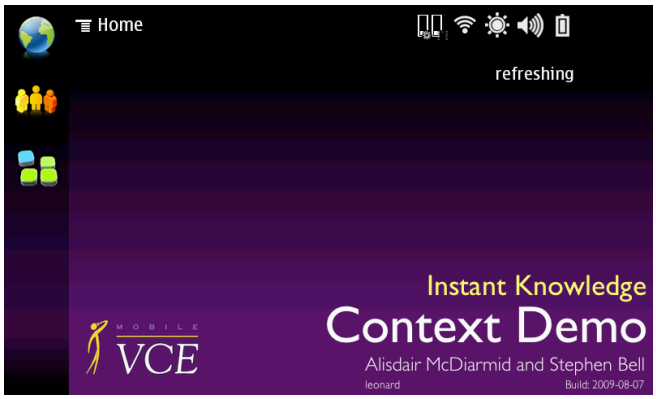

Figure 2: Desktop example image

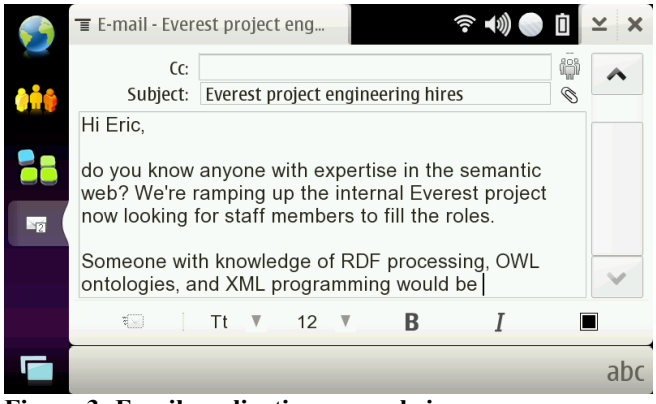

Figure 3: Email application example image

Figure 4: Chess application example image

The images from the Chess application had a lower compression ratio than the email or desktop, largely because there are no blocks of solid colour. For our application, the desktop and email client data sets are more representative and therefore more important.

Our results show that gzip was consistently the best performer in reducing the file size, although followed closely by the other algorithms. However, higher levels of compression resulted in a lower write frame-rate, with execution times for png and gzip especially high. The results show that fastlz with level 1-compression uses least CPU. This, along with an acceptable compression ratio for most data, led us to select this algorithm for use in the recording application.

	fastlz1	fastlz2	gzip1	gzip9	liblzf	png
Chess	61.82%	61.77%	42.81%	38.15%	52.65%	42.49%
Desktop	10.02%	9.50%	7.84%	6.03%	9.60%	8.19%
Email	5.64%	5.11%	4.60%	3.38%	5.65%	4.53%

Table 1: Compression ratio results

	fastlz1	fastlz2	gzip1	gzip9	liblzf	png
Chess	2.956	3.228	116.28	115.11	5.727	869.538
Desktop	1.41	1.828	65.246	59.019	2.31	138.604
Email	1.086	1.3	47.764	45.092	2.77	90.809

Table 2: Time to compress 20 images (seconds)

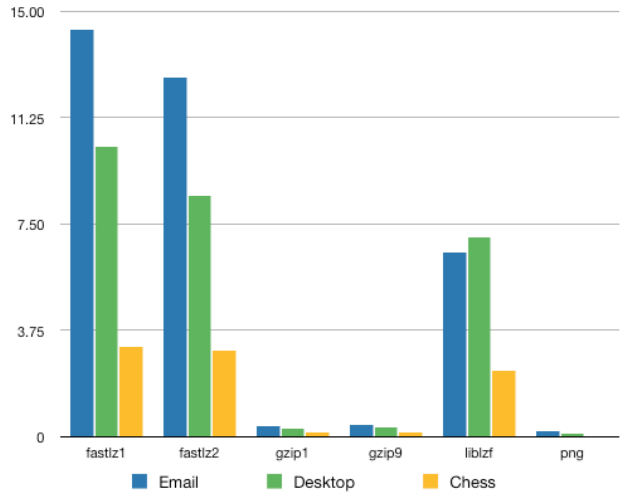

Figure 5: Calculated frames per second for several compression algorithms over three data sets

The frame period can be calculated by summing the time to compress one frame, and the time to write the compressed frame to disk. The average time to compress one frame is taken from the timing experiments, while the time to write a compressed frame was derived using the disk throughput, obtained from the time to write one raw frame to disk. The resulting frame periods were then converted to frames-per-second values and are displayed in Figure 5.

VI. User Interaction Session Playback

The recorded user interactions cannot be replayed on the portable device. Instead, the data is stored until the device is returned after the experiment, at which point it is copied onto a desktop computer and processed. This data includes the compressed screen captures, user input recordings, and associated device and user context information.

Figure 6 shows the initial implementation of the playback software. The user interface is intended to mimic a video playing application, with standard playback controls and a timeline slider. An interaction session is presented as a video, running in real time. The sequence of screen captures is animated continuously, touchscreen events and context data are presented at the appropriate times, and the user is able to scrub back and forth through the timeline to review interesting user interactions.

Recorded mouse events—touchscreen finger presses—are displayed as a translucent circle on the display. Keyboard events will be presented to the user through an animated graphical recreation of the device's keyboard. Context status is currently shown in simple text form in the interface: at present, only the position of the slide-out keyboard and the screen backlight status are shown, although there is a wide range of other device context available.

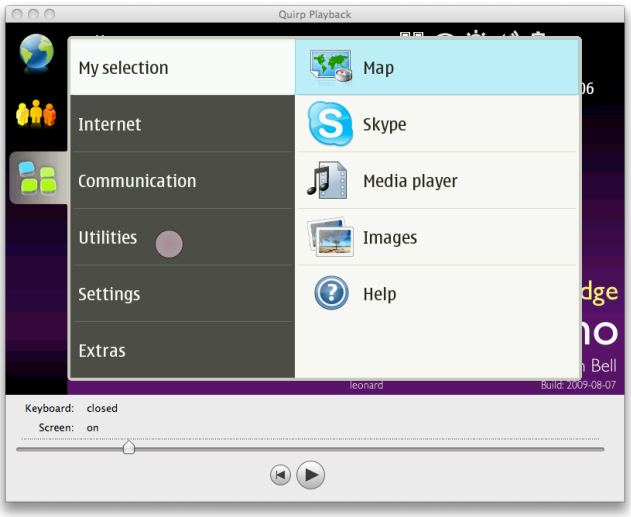

Figure 6: Playback software

VII. Future Work

Immediate work will focus on three things: optimizing the recorder to reduce processor and memory usage, improving the playback application, and incorporating more user context into the system.

An observation system running on the handheld device will have an impact on how the user perceives the device. It is therefore important to reduce the resource usage of the observation system. The application will be optimized using a special kernel image for the N810 allowing us to profile the application to see where the resources are being used.

The playback software will be improved by adding a keyboard representation to display keyboard events. The ability to single step through frames will enable more detailed analysis of the user interaction sessions. Better logging will profile a complete listing of the important parts of a user interaction session.

We intend to add more context information to the system as a whole. Initial investigations show that we can investigate numerous other pieces of context, including the battery level, whether there are headphones plugged in, and the ambient light level of the environment.

The device also has a GPS chipset, which may be useful in determining a user's location. A small accelerometer could be attached to the USB port and provide device acceleration readings, useful for determining how the user is moving in the environment. The playback software would be updated to display all of this information to the user.

There are also longer-term plans for further development of the concept. An initial approach of recording a screen and input user-interaction replay is useful in itself. One direction that we are just beginning to explore is augmenting this replay with context information, to allow a more detailed inspection of the device and user state at the time of each interaction. The IK context system will be modified to facilitate this, and new usability-evaluation tools must be created to navigate the information in a useful manner.

Since the storage capacity of the handheld device is limited, for long-term field trials, there must be a way of communicating the data off of the device to a remote store. This would allow the device to purge any of the previously recorded interaction sessions, and continue to record. This will be investigated in much more detail in follow-up work.

Finally, it is desired that the system be tested in the field, to verify that the system is robust enough to be deployed in real-world situations.

VIII. Conclusions

Handheld devices are becoming powerful enough to be used to record user interactions during usability evaluation studies. A technique for recording user interaction sessions by monitoring the inputs and outputs to a device has been proposed. This interaction recording method efficiently records the user's inputs to the device, and what is being output on the screen around the time of the interaction. The method and a prototype implementation of the recording software are described above. Immediate work includes optimization and performance evaluation of the prototype, and further work on the playback software. Future directions include expanding the amount of recorded context, a solution for long-term field studies on devices with limited disk space, and a usability evaluation in the field with test subjects to verify the robustness and usability of the software.

Acknowledgment

The work reported in this paper has formed part of the Instant Knowledge Research Programme of Mobile VCE, (the Virtual Centre of Excellence in Mobile & Personal Communications), www.mobilevce.com. The programme is co-funded by the UK Technology Strategy Board's Collaborative Research and Development programme. Detailed technical reports on this research are available to all Industrial Members of Mobile VCE.

References

[1] A. Stoica, G. Fiotakis, J. Simarro Cabrera, H. MuÒoz Frutos, N. Avouris, and Y. Dimitriadis, (2005), Usability evaluation of handheld devices, Proc. PCI 2005, Volos.

[2] Hummel, K., Hess, A., & Grill, T.. Environmental Context Sensing for Usability Evluation in Mobile HCI by Means of Small Wireless Sensor Networks, Proc. MMoM 2008, Linz.

[3] Roto, V. & Oulasvirta, A. (2005) Need for non-visual feedback with long response times in mobile HCI. WWW '05, 775–781.

[4] van Elzakker, C., Delikostidis, I., & Oosterom, P.. Field-Based Usability Evaluation Methodology for Mobile Geo-Applications. *The Cartographic Journal*; 45(2):139–149, 2008.

[5] Irvine, J., McDiarmid, A., Saunders, C., Tomlinson, A., & Jefferies, N. (2008) Instant Knowledge: Secure Autonomous Business Collaboration. In Wireless World Research Forum 20, 2008.

[6] Banford, J., McDiarmid, A., & Irvine, J. (2009) Relationship Mapping for Proactive Growth of Knowledge Networks. IEEE VTC 2009 Fall.

[7] Jakob Nielsen, *Usability Engineering*. Morgan Kaufmann, 1994.

[8] C. Benson, A. Elman, S. Nickell, C. Z. Robertson. *GNOME Human Interface Guidelines 2.2*. Available from: http://library.gnome.org/devel/hig-book/stable/feedback-response-times.html.en. Accessed: 11/09/2009.

[9] FastLZ – lightning-fast compression library. Available from: http://www.fastlz.org/. Accessed: 11/09/2009.

[10] Gzip: Open source command line data stream compressor and archiver. Available from: http://www.gzip.org/. Accessed: 11/09/2009.

[11] Liblzf. Available from: http://oldhome.schmorp.de/marc/liblzf.html. Accessed: 11/09/2009.