Published in: *Interacting with Computers*. (2013). 25(3), 242-258.

A Design, Tests, and Considerations for Improving Keystroke and Mouse Loggers

Jonathan H. Morgan[1], Chen-Yang Cheng[2], Christopher Pike[3], & Frank E. Ritter[1*]

[1]The College of Information Sciences and Technology
[3]Applied Research Lab
Penn State
University Park, PA 16802

[2]Department of Industrial Engineering and Enterprise Information
Tunghai University, R.O.C

* Corresponding author

~8300 words (body)
5 tables
14 figures

Draft of 15 October 2012, rui65.doc

jhm5001@psu.edu, chenyang@psu.edu, cuc174@gmail.com, frank.ritter@psu.edu
+ 1 (814) 599-5535

Abstract

We start by reviewing several logging tools. We then report improvements to a keystroke logger we have developed for the Mac and PC, Recording User Input (RUI). These improvements include changes to its interface, increased accuracy, and extensions to its logging ability. RUI runs in the background recording user behavior with timestamps and mouse location data across all applications—thus avoiding problems associated with video logs and instrumenting individual applications. We provide a summary and comparison of tests for loggers and and present procedures for validating logger timing that quantifies timing accuracy using an external clock. We demonstrate these tests on RUI and three other applications (Morae, Camtasia, and AppMonitor). We conclude by providing some general specifications and considerations for creating, testing, evaluating, and using keystroke and mouse loggers with respect to different experimental questions and tasks.

Keywords

User logs; Timing validation; Logging tools; Keystroke logs; Protocol data; Sequential data

Acknowledgements

This project was supported by ONR Contracts N00014-03-1-0248, #W911QY-07-01-0004, and N00014-11-1-0275. Bonnie John's experiences with and comments about RUI led to many of the developments described here. Geoffrey Morgan and Susan Strayer contributed to the development of PC RUI 2.04; Razvan Orendovici created the version of RUI that anonymizes keystrokes; Joe Sanford has helped maintain RUI; and Thomas George provided technical and procedural advice. Brad Best helped fit the gamma curve. Greg Plumb helped gather data. In addition, Olivier Georgeon, Wayne Gray, Jong Kim, Alex Kirlik, William Stevenson, Michael Wenger, and several anonymous reviewers provided comments that improved this paper.

1. Introduction

Obtaining and validating accurate timing for computer users has arguably become more difficult over time as operating systems have become more complex (Myors, 1999; De Clercq, Crombez, Buysse, & Roeyers, 2003). Naturalistic studies of computer use and human-computer interaction (HCI) studies, nevertheless, require accurate logging of keystrokes and mouse movements (Ritter, Kim, Morgan, & Carlson, 2013). Consequently, there is a wide variety of testing environments but few general logging tools.

In this report, we start by reviewing several logging tools. We then report improvements to a keystroke logger we have developed, RUI. These improvements include changes to its interface, increased accuracy, and extensions to its logging ability. We provide a summary and comparison of tests for loggers and new procedures for validating logger's timing that quantifies timing accuracy using an external clock. We demonstrate these tests on RUI and three other applications (Morae, Camtasia, and AppMonitor). We then examine these applications with respect to these tests. We provide some general specifications and some considerations for creating, testing, evaluating, and using keystroke and mouse loggers with respect to different experimental questions and tasks.

2. Review of keystroke loggers

Researchers recording behavior on computer interfaces for later analysis (e.g., Patton & Gray, 2010) confront several challenges. Existing solutions are either not general or limited in some way. For example, video recording methods will provide a record of working with nearly any interface, but they are cumbersome to use and their sampling rates (typically 24 times per second) are insufficient for measuring many of the smaller effects required by HCI's and psychology's paradigms, as noted by Plant, Hammond, and Whitehouse (2002), Plant, Hammond, and Turner (2004), and Alexander, Cockburn, and Lobb (2008)[1]. In addition, as Alexander et al. (2008) point out, video logs are vulnerable to interpretation errors when analyzing complex sequences or logic events (such as menu selection) because they depend upon the analyst's contextual knowledge.

[1] Video sampling rates constrain not only the ability to log user activity but also to validate it. Audio tests (with a sampling rate of approximately 8,000 samples/s as compared to a sampling rate of 24 samples/s for video) provide a powerful alternative.

When possible, direct instrumentation of the interface or task is quite useful because it allows analysts to specify and record the data necessary for their research in vivo. Many commercial applications, however, are designed to resist instrumentation for proprietary reasons.

Recording behavior with timing information directly from a keyboard or keypad using hardware dongles is another way forward (e.g., St. Amant, Horton, & Ritter, 2007). Dongles such as KeyDemon USB and KeyDemon PS2[2] are available to record keystrokes, but timing information at the millisecond level is just now being introduced in these commercially available solutions. Hardware approaches are typically very accurate; however, collecting data from these sources does pose some challenges, particularly for naturalistic or large geographically distributed HCI studies.

Where indirect instrumentation is possible, such as modifying a library that is used by many applications, it provides a useful and flexible solution. AppMonitor (Alexander, et al., 2008) provides yet another alternative, a light-weight client-side logger for Windows machines. AppMonitor records not only low-level actions but also the interface semantics of actions where these are defined. Like Alexander et al., we advocate the use of client-side loggers to capture user behavior not only because, as they have persuasively shown, such loggers can effectively capture interface related semantic content, but also because their relatively small footprint makes them resistant to rate errors and time stealing.

[2] These and other examples of hardware solutions can be found at *www.keelog.com*, while more current examples can be found by searching the Internet.

We note several software loggers in addition to RUI[3], including AppMonitor, Camtasia[4], Inputlog[5], Morae, MouseTracker[6], and DirectRT[7]. While these are all software solutions, it is useful to distinguish between task-oriented collection tools and stimulus presentation tools. The first category is primarily suited for data exploration and testing in naturalistic settings; the second for experiment development and hypothesis testing in lab settings. What primarily distinguishes these two groups is the degree of experimental control allowed by the tool upon the user environment.

The first group (e.g., RUI, AppMonitor, Camtasia, and Inputlog) provide little or no experimental control. RUI, AppMonitor, and Camtasia require initialization, while Inputlog requires users to select their recording preferences and identify the testing session. These tools are best suited for generally unstructured task environments, where investigators are primarily interested in identifying key features of the task or activity as users, computer user models, or robots experience it. In these instances, investigators may not yet have a strong theory regarding the behaviors in question but rather are performing exploratory analysis. Loggers, such as Morae, represent an intermediary position, as they do not structure the user's task environment but do depend on external recording equipment in a HCI lab setting.

---

[3] Since starting this project we have also found out about Fastlog by De Clercq (users.ugent.be/~adeclerc/fastlog) and Inquisit by Millisecond software (www.millisecond.com). Interested readers may wish to examine them as well.

[4]Camtasia and Morae are available at *www.techsmith.com/morae.asp* and *www.techsmith.com/camtasia.asp*.

[5] Inputlog is primarily used in writing research (e.g., Latif, 2008; Sullivan & Lindgren, 2006; Van Waes, Leijten, Wengelin, & Lindgren, 2011), and is available at *www.inputlog.net/*.

[6] MouseTracker (Freeman & Ambady, 2010) supports the spatial analysis, raw time analysis, and distributional analysis of mouse movements (but not keystrokes), and is available at *mousetracker.jbfreeman.net*.

[7] DirectRT is an experimental suite designed to help design and run cognitive and perceptual experiments, and is available at *www.empirisoft.com/directrt.aspx*.

The second group (e.g., MouseTracker, DirectRT, ePrime and perhaps Inquisit) allow experimenters to both structure and record activity using both stimulus presentations and analysis tools, providing them a high degree of experimental control. They, however, require experimenters to specify where and when the experimental stimuli occur, making them excellent for psychological studies of low-level stimuli but unsuitable for many naturalistic HCI or human-robot interaction (HRI) experiments.

We, here, are primarily concerned with task-oriented HCI and HRI tools and their validation. We thus limit our focus to two RUI platforms (Mac RUI and PC RUI), Morae, Camtasia, and AppMonitor, as these have all been used in HCI or HRI experiments. We first introduce Mac RUI and PC RUI as examples of a HCI/HRI logger. We then describe a comparative study of these loggers using five tests. We next discuss this study and our tests before addressing validation challenges and related risks associated with HCI and HRI loggers more generally. We conclude by providing a general specification for HCI and HRI loggers that arises from our experiences running HCI and HRI studies and designing HCI/HRI tools.

Before describing RUI or our tests in detail, a general introduction of our tested applications is in order. We begin with RUI. Recording User Input (RUI) is a pair of keystroke loggers that captures user behavior for the Mac and PC operating systems (Kukreja, Stevenson, & Ritter, 2007). We also examine two TechSmith applications, Morae, a usability testing package, and Camtasia, a recording and presentation tool. Though Camtasia is primarily used to record on-screen activity for making presentations and tutorials, its prevalence and accessibility have made it a popular alternative for recording user behavior. Finally, AppMonitor is a HCI logger, specializing in the capture

of semantic content by recording the movement between and manipulation of foreground and background applications.

3. RUI: Recording User Input

In their initial work on RUI, Kukreja et al. (2007) provided a general keystroke logger that worked on two major operating systems, Mac OS X (Mac RUI 1.0) and Windows (PC RUI 1.0). We briefly describe these versions, revisions, and some studies using them.

3.1 Mac RUI 2.0

Mac RUI 2.0 (January 2012) is a keystroke and mouse action logger implemented in the Carbon framework for Mac OS X. Users can record keystrokes, mouse clicks, and mouse movements, or any combination thereof. Logs, as noted in Table 1, include a header and a timestamp since log start, action, and argument (such as keys pressed or move locations) for keystrokes, mouse clicks, and mouse movements in a tab delimited file. In Table 1, the user moves the mouse, clicks, and then types, "Hello".

Table 1. An example log file from Mac RUI 2.0.

Subject ID:	C:\Documents and Settings\Desktop\Test 3.txt		
File Created:	11/25/2009	03:56:00 PM	
Elapsed [s]	Action	X	Y
0	Moved	275	605
7.322	Moved	276	605
7.486	Moved	278	605
7.655	Pressed	Left	
7.827	Released	Left	
8.486	KEY	SHIFT + h	
8.661	KEY	E	
8.994	KEY	L	
9.159	KEY	L	
9.319	KEY	O	
12.662	Moved	715	782

3.2 PC RUI 2.04

Written in the .Net framework using C#, the latest version of RUI for the PC platform, 2.04 (April 2012), records events more accurately than its predecessor (Kukreja, Stevenson, & Ritter, 2007) by generating a filewriting object from a list at the end of each recorded session and thus enabling it to smoothly capture events by generating a filewriting object to smoothly capture events across all Windows application at the end of each recorded session.. PC RUI 2.04 supports both HH:MM:SS and decimal time logging; it also collects summary statistics from the log file regarding keystrokes, mouse clicks, distances moved, and times[8]. We provide a very short example log in Table 2.

We also expanded the PC platform's logging capabilities by providing new runtime features such as the ability to note task changes using a task hierarchy, and the option to perform the NASA Task Load Index (TLX) test. The NASA TLX (Hart & Staveland, 1988) is a commonly used measure of workload. The workload factors assessed are: mental, physical, and temporal demands; performance; effort; and frustration. A user completes a ratings assessment for each task, as well as weighting each factor. Taking ratings with RUI provides a convenient way to record workload and ties these measures to keystroke and mouse click times by putting them in the log. The code to take the TLX measures adds about 80 KB to RUI's footprint.

[8] Distance is the movement by the mouse pointer in pixels, while the time is elapsed time from the initialization of the logger in seconds, unless otherwise specified by the user.

Table 2. An example log file from PC RUI 2.04.

```
Subject ID:        C:\Documents and Settings\Desktop\Test 4.txt
File Created:      10/26/2011   6:00:01 PM
Version:           2.04
Released:          07/20/2011

Elapsed Time [s]:  Action                  X          Y
0.390              Moved                   1279       504
0.406              Moved                   1278       504
1.468              Pressed Left
1.703              Key                                LShiftKey
1.828              Key                     H          Shift
1.921              Key                     E
2.031              Key                     L
2.125              Key                     L
2.296              Key                     O
2.421              Key                     OemPeriod
2.515              Pressed Right
2.640              Key                     D1
```

3.3 Previous uses of RUI

RUI's multiple platforms and timing resolution have proved a useful tool for a range of HCI and human-robot interaction (HRI) studies. Since its release, RUI has been used by us and others in HCI and cognition studies. Ritter, Kukreja, and St. Amant (2007) used PC RUI to log human and agent data for simple robot teleoperation tasks. Their work developed theories of HRI design and implementation, tested an ACT-R cognitive model, and demonstrated that a low-level cognitive user model could interact directly with an HRI interface to provide behavior predictions. These log files provided timestamps for mouse movements and keystrokes that were then compared to model predictions. Friedrich (2008) used Mac RUI to examine problem-solving strategies, while Kim (Kim, Koubek, & Ritter, 2007) and Paik (2009) used Mac RUI to study learning and retention. From this work, Kim developed cognitive task models that predict comparative retention rates of participants performing procedural tasks using keystroke commands versus using a GUI. In both cases, mouse movements and keystrokes were used to infer process within a task, with the total task time being derived from the logs.

Hurst, Mankoff, Dey, and Hudson (2007) used RUI to implement a pseudo-haptic technique for improving user interfaces for physically challenged populations. This tool uses RUI to capture where users click on the screen (using the (x,y) locations recorded in the log) to make the target areas bigger and more predictable. In an unpublished study, another group has used PC RUI 2.04 to record users and identify them for a cybersecurity application. Here, the relative timing between keystrokes was used. Drawing upon these and other experiences with RUI, we have revised both existing RUI versions, including versions that anonymize personal information while retaining data that ASCI keys were pressed.

4. Tests and standards for HCI and HRI keystroke loggers

We now discuss a comparative study of RUI's two platforms, two commercial keystroke loggers (Morae and Camtasia), and AppMonitor (Alexander et al., 2008). Table 3 summarizes the tests and standards applied to each application. The table organizes the rest of this section, because in each subsection we describe the rational of each test or standard, the method used, and the test results for each application.

This table generalizes the tests we used to develop RUI (Kukreja, Stevenson, & Ritter, 2006), as well as introducing tests used in the development of later versions of RUI and other loggers. Table 3 begins to note important characteristics for these instruments—that they are reliable and robust against external disturbances (although defining all disturbances is difficult) and load requirements, have (where necessary) small or well-understood biases, possess adequate precision to see effects of interest, and lead to replicable and thus relatable results. This list is not exhaustive, in that as a logger is used

in new ways further criteria and further artifacts are often identified. The list should be useful already, in that it provides a means of differentiating loggers.

Table 3. Tests and standards for HCI and HRI keystroke loggers.

1. No zero interkeystroke timings.
2. A gamma distribution of interkeystroke times.
3. A narrow distribution of mouse position logging times for fixed logging times that is comparable to the requested sample rate.
4. Comparison of logs with an external clock.
5. Low temporal drift over multiple days.

4.1 No zero interkeystroke timings

Except for a few unique and identifiable key combinations (e.g., CTRL, ALT, DEL), few or no zero interkeystroke time events should be present in the test data. A 0 ms interkeystroke timing event arises when two keystrokes have been logged simultaneously, indicating that either a batching error has occurred (two keypresses processed at the same time that were not pressed at the same time) or that the user has, in fact, pressed two keys essentially simultaneously. In previous validation studies (e.g., Ritter & Wood, 2002), loggers under a high enough load often failed this test when ran on multi-user systems. The analyst can differentiate between these two cases by examining the logs. The first case, the presence of batching errors, constitutes a fundamental but frequently correctable development error.

We have found 0 ms timings for mouse movement events. In these instances, the mouse appears to send two locations in the same sample period. This appears to be due to

the mouse hardware sending two locations, in order, within its same cycle period. To test this, we examined naturally generated logs, discussed in the proceeding section.

Method

Materials and Participants. The participants (n=6) for this test and the other tests were from a sample of convenience (colleagues not affiliated with the project). We recorded keystrokes and mouse clicks using a MacBook Pro 3.1 running Mac OS X v. 10.4.11 to test Mac RUI 1.0; and a Windows PC on a Dell Intel® Core™ 2 CPU to test PC RUI 2.04, Morae 3.0, Camtasia Studio 6, AppMonitor 1.2.1. We performed all keystroke and mouse trials reported here using a Logitech USB keyboard[9] and Logitech laser mouse respectively. We did, however, use the keyboards and mice accompanying each system during the pilot testing of the external clock test to confirm that the audio waveforms of keystrokes and mice do not vary significantly with respect to the physical hardware employed for the test. In later tests, we note changes to this basic setup and any additional apparatus if used.

Procedure. The participants were logged over three 30-minute trials over the course of two days for Mac RUI, PC RUI, Morae, and AppMonitor. In each trial, we used unordered combinations of the ASCII (a-zA-Z1-0, symbols) keys, modifier keys, mouse clicks, and mouse movements. We were unable to perform the test for Camtasia because it records events into a WAV movie file; we could only compare automatically generated sounds for each keystroke or mouse click with physical mouse clicks or keystrokes we recorded for the external clock test.

[9] The USB polling rate is 125 Hz, introducing an average delay of 4 ms between the execution of a keystroke and its acknowledgement by the OS.

Results

For Mac RUI, PC RUI, Morae, and AppMonitor, we found no 0 ms interkeystroke timing events for samples of approximately 10,000 events for each—the recording of mouse movements caused the sample totals to differ slightly. Our test of Camtasia was inconclusive. While we found that each audio representation corresponded with a physical sound (120 keystrokes), the log lacked the granularity necessary to detect batching errors.

4.2 A gamma distribution of interkeystroke times

Reaction times typically follow distributions with a single long right tail because reactions times can never fall below 0. Consequently, distributions such as a gamma distribution (Luce, 1986; Patton & Gray, 2010; Van Zandt, 1995; Van Zandt & Ratcliff, 1995), a log-normal (Braun, Rousson, Simpson, & Prokop, 2003), or an ex-Gaussian (Van Zandt, 2002) generally characterize test data. The absence of such a distribution suggests that either the data is not reaction time data or that the logger is not recording correctly.

Method

We analyzed the data we tested for zero interkeystroke timing events to determine the distribution of interkeystroke times.

Results

The keystroke times for Mac RUI and PC RUI when fitted to a gamma distribution exhibited r^2 values of .977 and .963 respectively. Fig. 1 shows the distribution and fit for PC RUI. We, however, could not perform this test on AppMonitor, Camtasia, or Morae. For AppMonitor, we could not replicate a 30 min. typing trial because AppMonitor only

records timestamps for two ASCII keys, the Enter key and spacebar. As in the previous test, Camtasia's dependence on WAV files made this test impossible. As for Morae, its exported logs only provide timestamps in seconds, making the test meaningless[10].

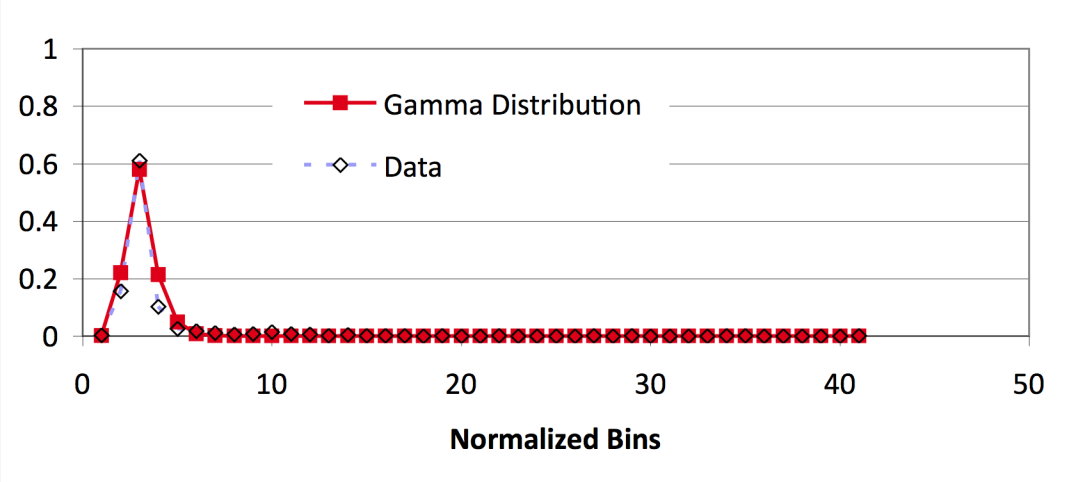

Fig. 1. Reaction time data: PC RUI 2.04.

4.3 A narrow distribution of mouse position logging times for fixed logging times

Whereas mouse clicks and keystrokes have a definitive stopping and starting point, mouse movements are a relatively fluid activity. Defining a meaningful mouse move is a difficult heuristic that most designers of logging software avoid. Instead, designers frequently choose to log changes in the mouse's position, and thus are obliged to rely upon the default refresh rate of an event listener (RUI is in this group). Logging so many events can contribute to memory delays that in turn generate statistical outliers in the timing data.

If a logger is recording mouse movements properly, the positions will be logged at a consistent interval (Dasarathy, 1985). This interval represents the update rate of the logger's mouse event listener. Loggers frequently use separate event listeners for

[10] Using Morae Manger's "Search Results" window and Audacity, you can generate by hand a log with ms timing. We did this for the external clock test; however, this method is prohibitive for tests consisting of more than a couple hundred events.

recording keystrokes and mouse activity. In either case, the update rates generally range between 15 and 20 ms. An application's update rate establishes the limit of its accuracy. Thus, by examining the distribution of mouse movements throughout a trial's time course, one can determine the range of this interval, and thus have a general sense of an application's accuracy in this regard. If the logger uses mouse events (as opposed to polling), then this test determines if the times are reasonable (intervals between 15 and 20 ms) and that there are no artifacts present in the timing data.

Method

To verify the distribution of mouse position logging times, we conducted three 3 min. sessions. We, however, could only obtain mouse move events for Mac RUI 1.0 (6,522 events), PC RUI 2.04 (11,336 events), and AppMonitor (12,296 events)—the number of events varied because each session was a free movement or scrolling test.

Testing AppMonitor required us to scroll the mouse continuously during the course of the trial because AppMonitor only records timestamps for mouse move events associated with scrolling. Thus, while we expect that AppMonitor is accurate, our claims regarding AppMonitor are weaker because of the limited nature of the test. We were not able to analyze Morae or Camtasia in this way. While Morae supports the video playback of mouse moves, it does not log mouse move events in either its "Search Results" window or in its exportable logs. Camtasia does not log mouse moves. A mouse could start and stop, creating long gaps between mouse location recordings. So, in our tests, we discarded events that had a gap between moves of over 20 ms.

We based our choice of 20 ms on an analysis of mouse move distribution times across multiple trials. These distributions were consistent across trials and platforms. Fig. 2

shows a time interval for one such trial. In Fig. 2, the first distribution of times between mouse locations has a break point at 20 ms, very close to the expected sampling rate of 18 ms. There are very few moves above 20 ms. A single pixel movement generates a mouse event, so above 20 ms per pixel the mouse is moving less than 50 pixels per second, or about 0.5 inches per second. These cases, however, are not stressing the logger to pick up times. This number could be adjusted; the histogram shows that the slower moves, unless there were many of them, would not change the logging rate and would, most likely, be indistinguishable from times between moves.

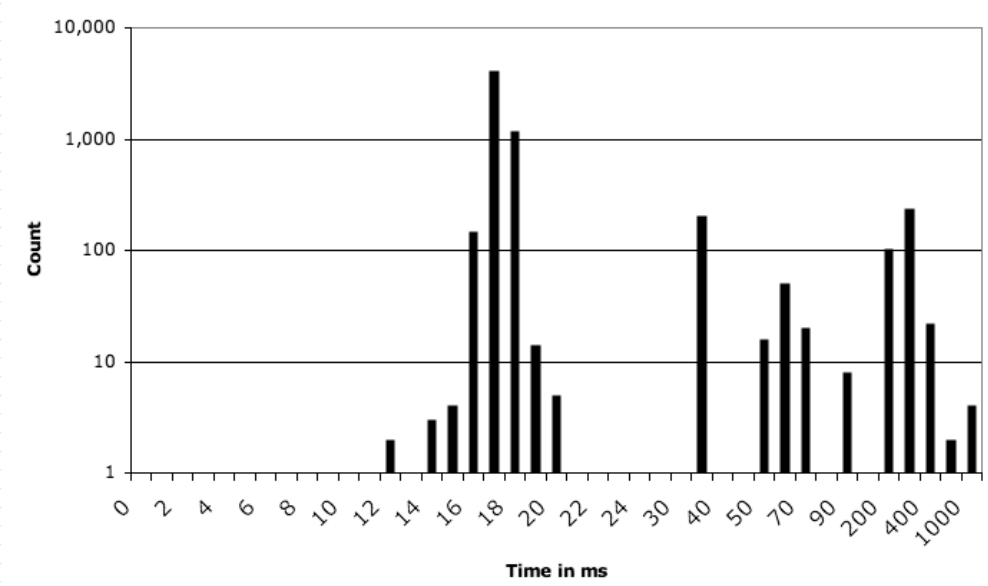

Fig. 2. Log histogram of times between mouse locations from Mac RUI 1.0 for 5.5 min of moves (N=6,514 data points). Note that the bins are non-linear to make the plot more readable.

Results

After adjusting the data as noted above, the remaining moves for Mac RUI (5,486 events), PC RUI (9,740 events), and AppMonitor (12,290 events) exhibited modal values of 16 ms for Mac RUI and AppMonitor and 17 ms for PC RUI between logging events.

Thus, the mouse logging behavior was not perfect, but does appear to neither over nor under sample the mouse move locations.

4.4 Comparing the keystroke logger's time with an external clock

Recording timestamps within multitasking operating systems introduces complications. Routine synchronization via NTP servers does nothing to correct low-level, short-term time stealing (variance in timestamps due to logging processes not getting their allotment of time, MacInnes & Taylor, 2001) or its effects (i.e., rate errors). While built-in time synchronization programs such as those found in Mac OS 8.5.1 and later, the prevalence of synchronization and calculation software, and the growing use of special purpose Window classes has generally made computer clocks more reliable, there remains a need for tests that are not only resistant to but also capable of detecting time stealing. We now describe one such test and its limitations. (De Clercq et al., 2003, used a more direct, but less available, hardware-based solution to perform a test like this.)

Camera-based tests: Method

We used basically the same approach with only changes to the apparatus and procedure.

Apparatus. For the external validation, we used a *Robic SC-505 5-Lap Memory Stopwatch* (accurate to a 1 ms/day) and a Canon DVD Camcorder DC220. For the analysis, we used Ace DVD Audio Extractor®[11] and Audacity®[12].

Procedure. Keystroke and mouse click samples were collected from each application (Mac RUI 1.0, PC RUI 2.04, Morae 3.0, Camtasia Studio 6, and AppMonitor 1.2.1) while

[11] *www.freedownloadscenter.com/Multimedia_and_Graphics/Misc__Sound_Tools/Ace_DVD_Audio_Extractor.html*

[12] Audacity 1.2.6, *audacity.sourceforge.net*

being recorded on the camcorder using its internal microphone. For each application, we compared the audio waveform of the physical action (e.g., key pressing, mouse clicking, or stylus tapping) to the logger's output. During each trial, the participant waited approximately two seconds between each event (keystroke or mouse click depending on the test) and after every ten events, moved the mouse to delineate between sequences of events[13].

We controlled for the bias introduced by sound propagation by keeping the distance between the camera's microphone and the keyboard or mouse less than 6'' at any point during the trial, meaning that the maximum degree of bias introduced by sound propagation across this distance was approximately a fixed 4 ms[14] (in most cases far less). An offset of 4 ms is within the variation of ± 20 ms found with this method. To reduce the effects of ambient noise, all trials occurred in a quiet room.

Determining the external clock's rate of drift. For our time trials, we used a camera capable of writing to DVDs as our external clock. To validate the camera, we performed two tests. We ran five trials where we recorded the National Institute of Standards and Technology's (www.time.gov/timezone.cgi?Eastern/d/-5/java) atomic clock for 35 minutes, the duration of the camera's DVDs. From these trials, we derived an average drift of 1 second per 804 s (the camera lags NIST). We further substantiated this finding by recording a high performance stopwatch for five trials and comparing the results of these trials with those of the earlier NIST trials. We found that the results were

[13] The spacing between keystrokes and mouse clicks ensured a clean audio waveform while the mouse movement between sequences allowed visually confirming starting and stopping points. In addition, these periods of mouse movement allowed observing if any logging errors occurred when shifting between event types.

[14] At 20 °C, sound travels 343 m/s. Therefore, at a distance of 1.37 m (approximately 4.49 ft), there will be a delay of approximately 4 ms.

consistent. Though this difference indicates a small drift in the camera's time, we were able to adjust all the subsequent data to account for this effect. Furthermore, the significance of the camera drift is mitigated by the fact that we are primarily interested in the range of the variation of the trial data with respect to an external clock rather than their absolute values.

Analysis. We extracted audio files from the DVD recordings using the Ace DVD Audio Extractor 1.1.2[15]. We then used Audacity® (1.2.6, audacity.sourceforge.net), an open source audio editor and recorder, to display the recording's audio waveform. Fig. 3 is an example of a keystroke waveform.

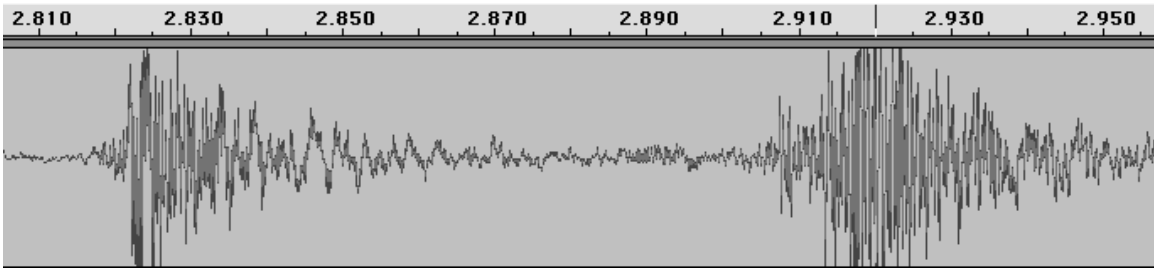

Fig. 3. Keystrokes in Audacity waveform. Time (x-axis) is in s and volume (y-axis) is in dB.

We verified that each wave was in fact a keystroke or mouse click by using Audacity®'s playback function[16]. Once verified, we determined the time of each keystroke or mouse click by recording the time of each wave's crest[17].

[15] *www.freedownloadscenter.com/Multimedia_and_Graphics/Misc__Sound_Tools/Ace_DVD_Audio_Extractor.html*

[16] In longer samples, visual confirmation was necessary. Visual checks are also useful when differentiating the sample's starting point from the application's activation sequence.

[17] There are multiple wave crests; the maximum of the first burst is used as the time. Fig. 4 has this indicated in Audacity at 2.822 s. We found this pattern to be consistent across varying keyboard and mouse types. Mouse clicks present a more pronounced but similar pattern. In all cases, the maximum of waveform has a maximum duration of 3 ms.

Limitations of a camera-based test. Though we have attempted to create a test based on an external clock, we have had to suffice ourselves with comparing application logs with extracted audio files from video recordings. The limitations imposed by this approach permit us to validate an application's accuracy only down to the 20 ms level after adjusting for the biases introduced by the camera's rate of drift, the distance between the camera and the keyboard or mouse, and the polling rate delay. Furthermore, we found that interpreting the data required us to create rather syncopated samples, approximately one to two seconds between each keystroke or mouse click. Though we have done trials with higher frequency inputs, we found that the sound waveform in these trials was too complex to interpret. Finally, this method allows us to validate individual keystrokes or mouse clicks, not specific mouse movements. Verification of mouse movements is impossible using this method because the analyst must align the sound of the keystroke or mouse click from the waveform with the tested application's log file—mouse moves do not make sounds. It does, however, provide a test that is not a binary pass/fail, but rather provides a measure of accuracy.

Camera-based tests: Results

Alignment of log files and the external clock. Fig. 4 shows an example of the time course of the alignment of the external clock times (x-axis) and the RUI times (y-axis) for 20 keystrokes and mouse movements for Mac RUI 1.0.

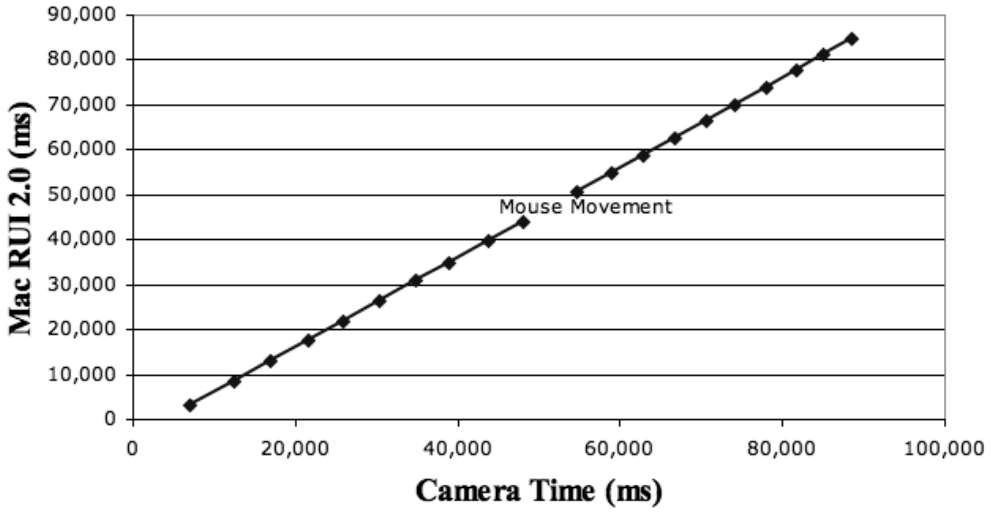

Fig. 4. Extracted external clock times vs. Mac RUI 1.0's recorded times for keystrokes.

Mac RUI results. The correspondence in Fig. 4 is a straight line, indicating a one-to-one correspondence between the times recorded by the external clock and those logged by RUI[18]. Fig. 5 shows the relative time differences (differences in the recorded offsets) over the course of 19 keystrokes, or approximately 1 min of activity. After comparing the times generated by the external clock to the RUI times, we found that the difference values vary from -27 to +15 ms over the course of the trial. Most measures are within ± 20 ms, and all are within ± 30 ms. Thus, these results indicate that timing of Mac RUI is not guaranteed at the 1 ms level, as previously claimed (Kukreja et al., 2007), but 30 ms would be a more appropriate accuracy. This result, nevertheless, suggests that this approach's can provide more tests to 10 ms or perhaps 5 ms resolution—as we performed these analyses we found that it was difficult to know the start of the keystroke time within 3 ms because of the shape of the waveform.

[18] Mac RUI records both upstrokes and downstrokes while the other applications do not. In other applications, we analyzed the downstroke of each keystroke.

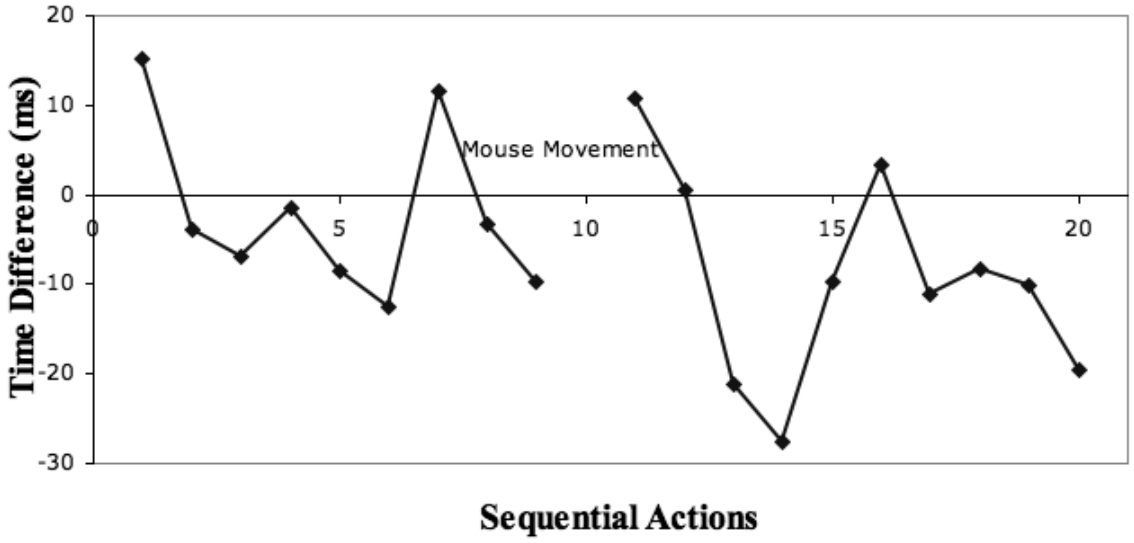

Fig. 5. Observed differences between RUI (Mac RUI 1.0) and the external clock.

Fig. 6 shows the differences for the 20 mouse clicks (39 data points)[19] taken from Mac RUI 1.0. We generally find that the degree of variation for these trials is less than for the typing trials, perhaps partly because mouse clicks possess a crisper audio signature, making the data interpretation easier. These measures show that the external clock generated times and RUI times appear to differ by -8 to +3 ms over the course of the trial. Most measures are within ± 10 ms, and all are within 11 ms. As in typing trials, the limitations of this approach prohibit us from certifying Mac RUI 1.0's mouse click data at the 1 ms level, but 20 ms accuracy may be appropriate. The degree of variation for these trials is less than for the typing trials, perhaps partly because mouse clicks possess a crisper audio signature, making the data interpretation easier.

[19] Mac RUI records both up and downstrokes, thus the 40 data points. We used the first event to synchronize the two times. Consequently, this data point does not appear in Fig. 6.

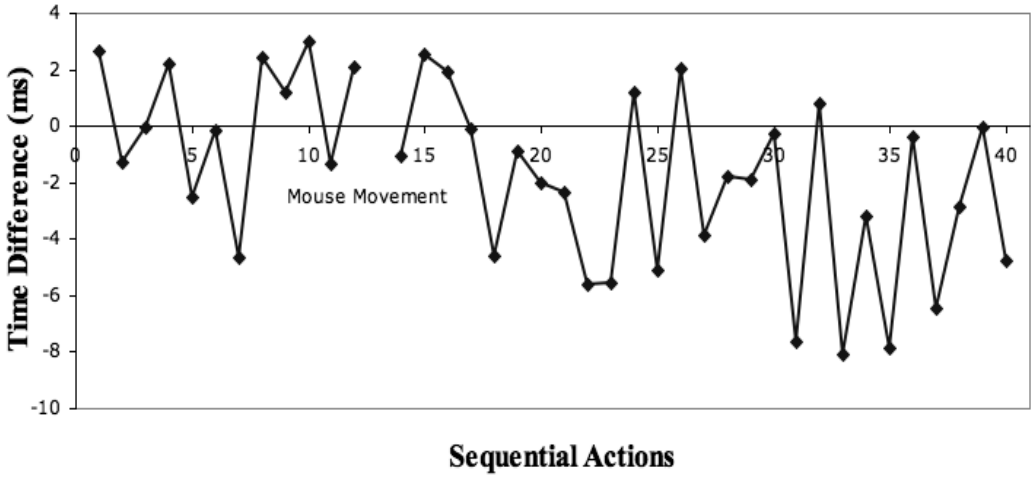

Fig. 6. Observed differences with the external clock for mouse clicks (Mac RUI 1.0).

PC RUI 2.04 results. Fig. 7 shows the differences between PC RUI 2.04 and the external clock for 20 keystrokes. These measures show that the external clock times and RUI times appear to differ by a range of -21 to +7 ms. Again, most points are within ± 20 ms. Nevertheless, based on this validation approach, we cannot guarantee RUI's timing at the 1 ms level, but suggest a rating of ±30 ms for keystrokes.

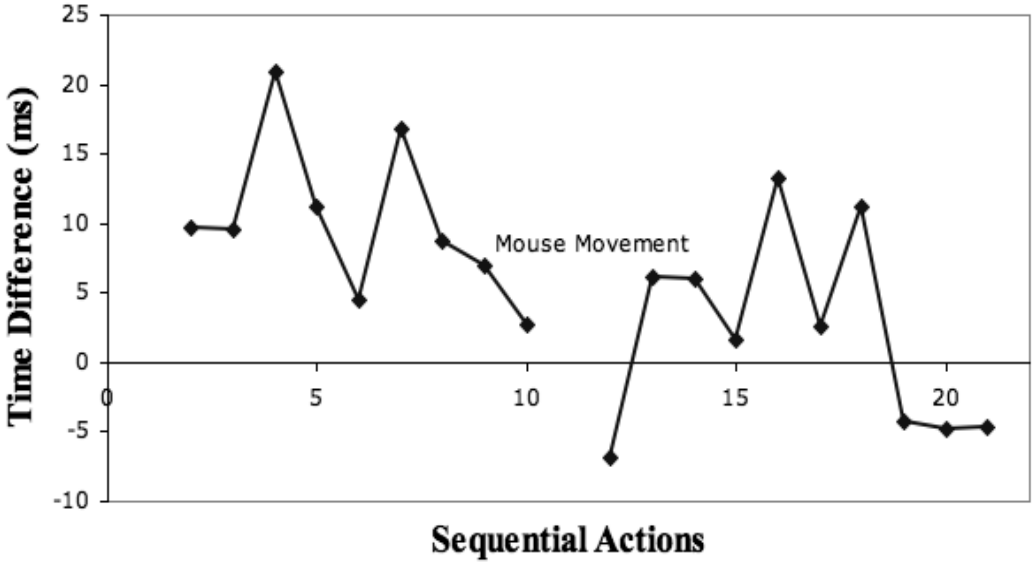

Fig. 7. Observed differences between keystrokes (PC RUI 2.04) and the external clock.

Fig. 8 shows the differences between PC RUI 2.0 4 and the external clock for 20 mouse clicks. These measures show that the external clock times and the RUI times appear to differ by -12 to +3 ms over the course of the trial. Most measures are within ± 10 ms, and all are within 12 ms. Thus, we suggest PC RUI is accurate to ± 20 ms for mouse moves.

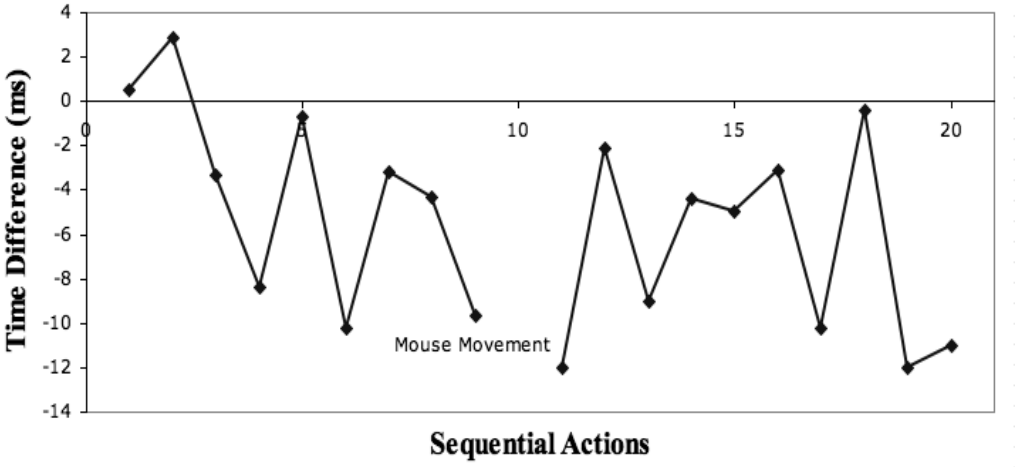

Fig. 8. Observed differences for mouse clicks (PC RUI 2.04).

Morae 3.0 results. Fig. 9 shows that the difference between the external clock and the times found in Morae Manager's *Search Results* window—these times provide more precision than Morae's default logs but are not exportable and cannot be cut and pasted. For both keystrokes and mouse clicks, Morae's exported logs use HH:MM:SS.S format that truncates the logged times to a tenth of a second. When using these logs, we could only obtain Morae's timing to ± 0.1 s precision. When recording keystrokes using results from the *Search Results* window, the corresponding times varied from -52 to +22 ms. The alignment between times derived from the external clock and Morae's logged times showed no noticeable drift or noise. These findings suggest that Morae is accurate to 100 ms (perhaps 60 ms), making it sufficient for most HCI usability studies.

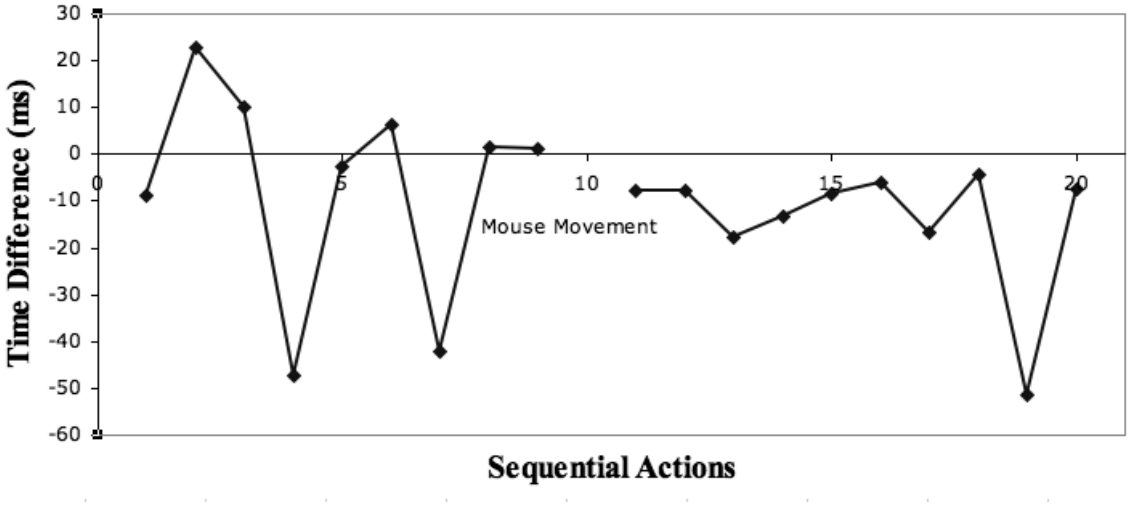

Fig. 9. Observed differences for keystrokes (Morae 3.0).

Fig. 10 shows the differences between the external clock and log times, recorded by hand, from Morae Manager 3.0's Search Results window, for 20 mouse clicks. These measures show that the external clock times and the Morae times appear to differ by -11 to +58 ms over the trial's time course. These findings indicate that Morae's accuracy seems to be consistent across modalities (±60 to 100 ms for keystrokes and mouse clicks depending on how conservative you want to be).

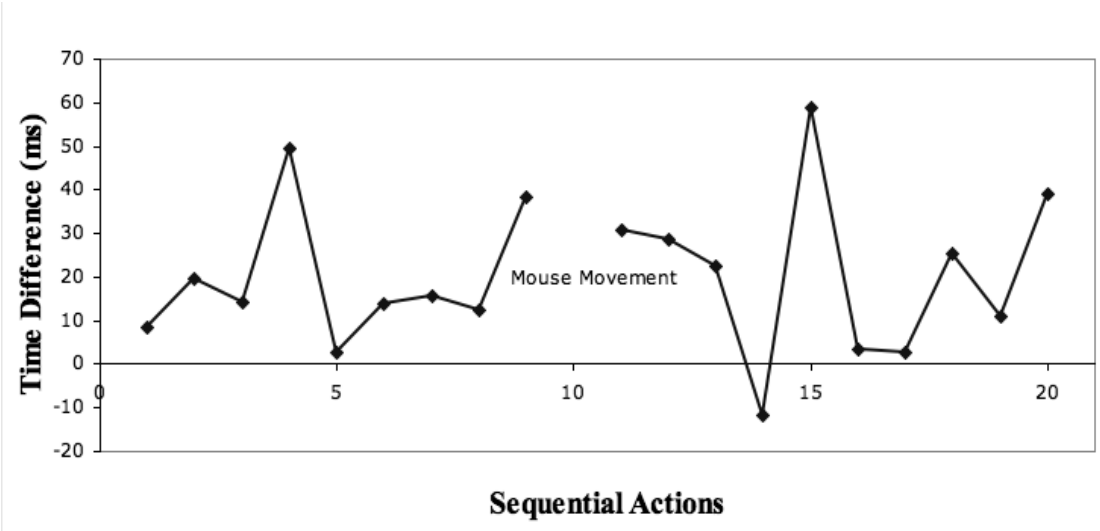

Fig. 10. Observed differences for mouse clicks (Morae 3.0).

Camtasia, Studio 6 results. Camtasia does not generate logs; rather it stores information in WAV and AVI files. Because Camtasia does not assume (in its default and commonly used setting) a microphone, it inserts a keystroke sound into an audio file for each keystroke entered. In these trials, we generated Camtasia's recorded times by analyzing the WAV file for each trial using Audacity®, as well as Camtasia's own audio analysis tool. We preformed an audio analysis because like other video logs the sampling rate is too low to record user behavior at or below 20 ms. We again found that the alignment for both keystrokes and mouse clicks was generally good without any noticeable drift or noise.

Fig. 11 shows the differences between times taken from Camtasia's audio recording and the external clock for 20 keystrokes. These measures show that the times extracted from Camtasia's WAV file when compared to the external clock, display a linear drift of approximately one second per minute, an error of about 1%. Thus, while the interkeystroke and mouse click times may be fairly accurate, we found this rate of drift held constant for ten trials. One feature of Camtasia may contribute to this relatively high rate of drift. Camtasia does not record the keystroke sounds but rather generates a very clean audio representation when each key is pressed. This is desirable for creating understandable audio tracks for presentations. On the other hand, it may be rounding off the sound times for each click by fitting them to the video's frame rate. So, one possible explanation of this drift is the effect of these accumulated rounding decisions.

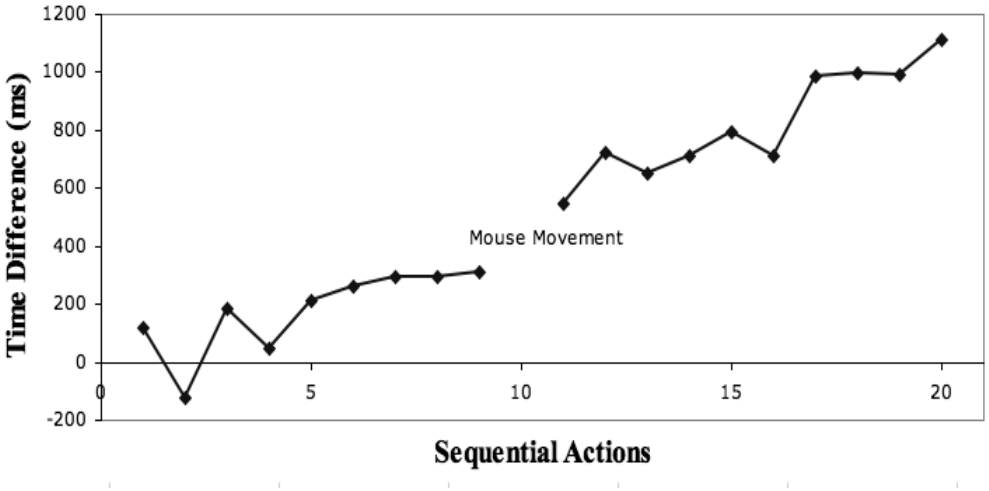

Fig. 11. Observed differences for keystrokes (Camtasia, Studio 6).

Fig. 12 shows the differences between Camtasia and the external clock for the 20 mouse clicks. These measures show that the external clock's times and Camtasia's times for mouse clicks also display a linear drift of approximately one second per minute. As in the case of keystrokes, Camtasia generates an audio representation of each mouse click, making mouse click timing susceptible to the same potential problem. Thus, we believe Camtasia is accurate to the nearest second.

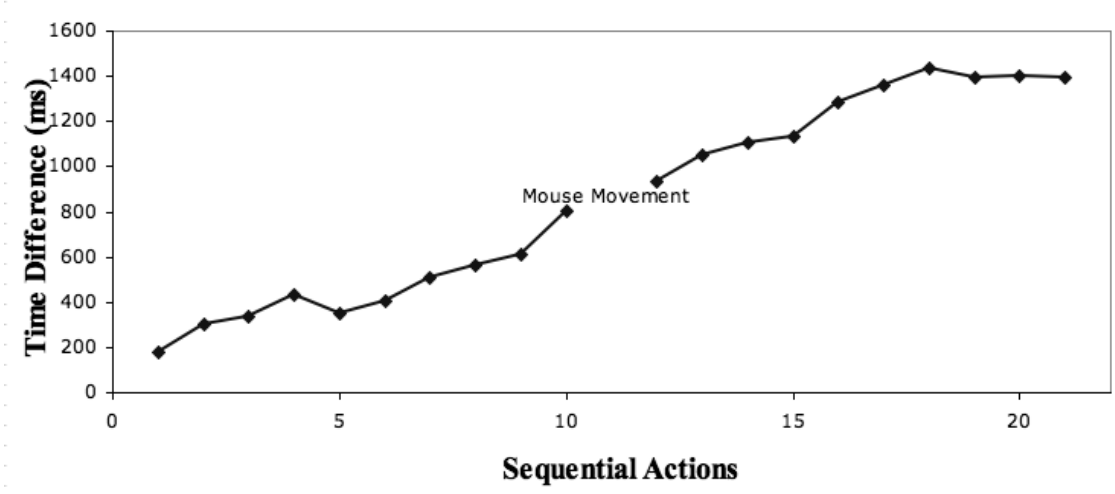

Fig. 12. Observed differences for mouse clicks (Camtasia, Studio 6).

AppMonitor 1.2.1 results. In our tests, we found AppMonitor to be fairly accurate. Each data point in Fig. 13 represents one instance of the enter key because AppMonitor only provides key names for the enter key and spacebar. Fig. 13 shows the differences between AppMonitor and the external clock for 20 keystrokes. These measures show that the external clock times and the AppMonitor times appear to differ by a range of -7 to +22 ms over the course of the trial. Like RUI, most points are within ± 20 ms of the external clock.

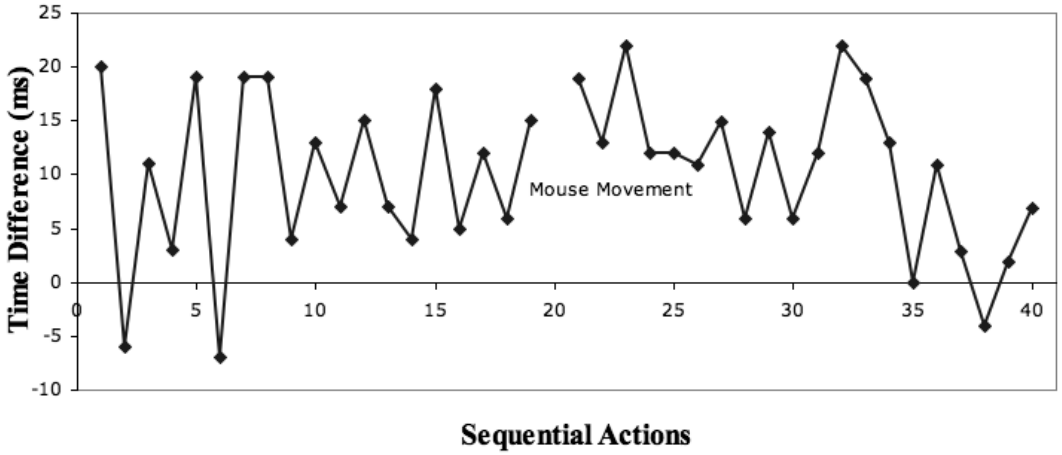

Fig. 13. Observed differences for keystrokes (AppMonitor, 1.2.1).

Fig. 14 shows the differences between AppMonitor and the external clock for 20 mouse clicks (39 data points). AppMonitor records both up and downstrokes. These measures show that the external clock times and AppMonitor's times appear to differ by -9 to +85 ms over the course of the trial. Most measures are within ± 10 ms. Consequently, AppMonitor appears accurate to 100 ms, but it might be better.

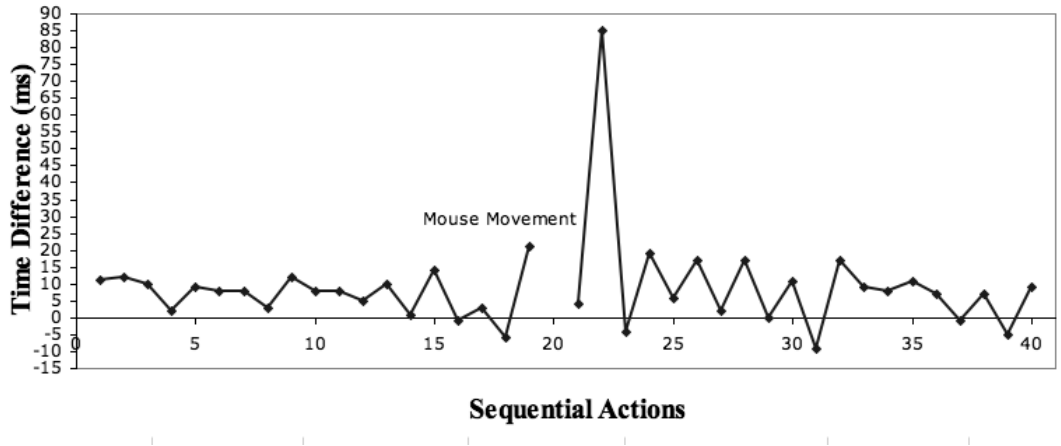

Fig. 14. Observed differences for mouse clicks (AppMonitor, 1.2.1).

External clock test summary. The camera-based tests are expensive to perform because they have to be performed by hand, but appear to provide a relatively high resolution test that can provide time estimates within 3 ms based on sound waveforms. These tests show that all loggers can record a set of timestamps within 3 ms of camera time, but that the loggers have between 10 and 80 ms variance in how well they record.

4.5 Low temporal drift over multiple days

Keystroke and mouse action loggers should exhibit low temporal drift over multiple days when run in isolation. Assessing an application's rate of drift over days and weeks, however, is surprisingly difficult. Determining an application's rate of drift requires knowing the external measure's rate of drift, as described in our discussion of the external clock test. With that knowledge, the analyst can then compare the application's timing with respect to an external clock, and thus determine to what degree this error influences the data.

This relatively straightforward description is complicated when one considers the constraints imposed by the external clock. We determined each application's rate of drift,

but it was a rough estimate limited by the recording time (30 min.) of our camera's DVDs. This limitation obliged us to record four 30 min. trials over the course of two days for Mac and PC RUI. We could not perform this test on Morae, Camtasia, or AppMonitor. For Morae and Camtasia, generating a (ms) log by hand at these time scales is prohibitive. In the case of AppMonitor, the fact that AppMonitor provides timestamps for a limited number of keystroke and mouse movements makes recording the breadth of naturalistic activity impossible, severely limiting this test's external validity.

Apparatus. We used a Canon DVD Camcorder DC220 to record the NIST atomic clock, and a Robic SC-505 5 Lap Memory Stopwatch (accurate to a 1 ms/day) to confirm the measurements.

Procedure. Over the course of a week we compared the times of Mac RUI and PC RUI with the NIST atomic clock times in 30 min. increments. We isolated these applications from NIST clocks for two days, resulting in 4 tests for each application. In each case, we adjusted the results to compensate for the external clock's offset, and confirmed them by using the stopwatch.

Results. For Mac and PC RUI, we found no perceptible lag during the 30 min. tests over the course of two days. We expect a drift could emerge for tests over multiple days or weeks. We believe these results have important implications for larger longitudinal studies that care about differences of seconds over days (which might be relatively rare), particularly as the size of the logger footprint increases to capture semantic data. On the other hand, lag occurring over multiple days is less important for hour-long studies, or studies that do not need accuracy within seconds over days or weeks.

5. Discussion and Conclusion

In this paper we have discussed risks to accuracy that confront experimenters and designers in capturing and analyzing more complex sets of interaction data. After briefly discussing RUI and its research applications, we described a comparative study of RUI, two commercial applications (Morae and Camtasia), and a logger from a university lab (AppMonitor). While we found that we could validate the logger's timing accuracy, including RUI's timing to ± 30 ms for keystrokes and ± 10 ms mouse moves, we also found that validating loggers for mouse moves and tests exceeding 30 min. was difficult.

We next summarize the results of these tests. We then examine what these tests mean for future loggers by creating a preliminary set of considerations for loggers meant to identify and address some of the major risks present in HCI and cognitive psychology studies related to keystroke loggers. We conclude with a final thought about future work and how these considerations highlight some current limitations of loggers.

5.1 Summary of results

The results of our tests are summarized in Table 4. They show measures on a range of loggers and that the revised and extended versions of RUI pass a wider range of tests than previously reported (Kukreja et al., 2007). These revised versions of RUI appear to record keystrokes and mouse clicks without batching, pass the gamma distribution test, accurately record mouse movements, and can accurately record keystrokes and mouse clicks with an accuracy of 30 ms, or slightly better in one case (Mac). This accuracy enables RUI to support a wide range of HCI and cognitive psychology tasks, but not all tasks. Finally, we did not include the results of test five (determining long-term temporal

drift) because we could only test the RUI platforms, and even for these platforms additional testing would be necessary to solidify our findings. We, however, included this test because we believe the increased prevalence of online HCI studies warrants considering long-term temporal drift and ways of testing it.

Table 4. *Summary of tests across applications.*

	Tests	Mac RUI 1.0	PC RUI 2.04	Morae 3.0	Camtasia, Studio 6	AppMonitor 1.2.1
1	0 Interkeystroke Timing Test	Pass	Pass	Pass	Fail[1]	Pass
2	Gamma Distribution Test for Keystrokes	.977	.963	NA[2]	NA[1]	NA[5]
3	Mouse Move Distribution Test Modal Values	17 ms	16 ms	N/A[4]	N/A[4]	16 ms
4a	Assessing Accuracy Using an External Clock for Keystrokes (in ms)	-28 to 15 (± 30 ms)	-7 to 21 (± 30 ms)	-52 to 23 (± 60 ms)	-123 to 1115 (~ 1 s)	-7 to +22 (± 30 ms)
4b	Assessing Timing Accuracy Using an External Clock for Mouse Clicks (in ms)	-8 to 3 (± 10 ms)	-12 to 3 (± 20 ms)	-12 to 59 (± 60 ms)	176 to 1480 (~ 1 s)	-9 to +85 (± 100 ms)

1 – Camtasia does not provide keystroke logs.

2 - Morae's exported logs only provide keystrokes to the nearest tenth of a second.

3 – AppMonitor does not provide timestamps for ASCII keys, with the exception of the Enter key and spacebar.

4- Morae and Camtasia do not log mouse moves at (time, x, y).

5 - Because AppMonitor only provides timestamps for the Enter key and space bar, test samples collected from AppMonitor are unlikely to representative, as it is impossible to fully replicate.

These tests can also be used to test and describe other loggers. Morae and Camtasia, for example, can accurately record keystrokes and mouse clicks occurring at time scales