A Survey of Windowing Environment Usage Patterns

Michael J. Katchabaw, Hanan L. Lutfiyya, and Michael A. Bauer
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract

To develop better computing environments, there is a need to know how these environments are being used. In this paper, we present a study examining how users make use of windowing environments to obtain this kind of information. The requirements of this study are discussed first, followed by the design and execution of the study, and finally the results of the study. This paper concludes with a summary and discussion of possible future directions.

1 Introduction

The use of computers has permeated practically every aspect of society. They are now in use in business, education, government, and homes; used for productivity, commerce, instruction, research, and entertainment. As we become increasingly dependent on computing in our day-to-day lives, the demands placed on our computing environments increase as well. To develop and construct computing environments which better meet the needs of users, we need to understand how users interact with and make use of these environments.

Users interact with their computing environments through some kind of interface. As time passed since computers were first put into use, technology improved and become considerably cheaper, allowing user interfaces to become more attractive, intuitive, and easier to use. For several years now, the interface of choice for most users of computer systems has been the graphical windowing environment, providing a reasonably simple yet effective metaphor for interacting with user applications. A window is simply a display area on a computer screen, usually rectangular in shape, used both to display data from an

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

application to its user, and to accept data from the user for the application. The graphical nature of these environments is a substantial improvement over their largely textual ancestors, both in terms of ease of use and overall appeal.

As applications execute in this kind of environment, windows are created full of buttons, menus, lists, boxes, pictures, text, and other elements to provide an interface to the application user. Since every user application can create multiple windows, and most modern operating systems allow users to run multiple applications at once on the same system, this can result in a very cluttered screen quickly. Through careful juggling and manipulation of windows, however, users can manage and work with multiple applications at the same time quite effectively. Window juggling can be accomplished in a variety of different ways. A desktop metaphor is a commonly used and often helpful way of thinking about this [5]; with the computer screen being a desktop, and the various windows on the screen being pieces of paper on the desktop. (In fact, this metaphor is so common that people often refer to the collection of windows and background from their windowing environment as the "desktop".)

- As windows are created by applications and used, windows that are not the current focus of interest tend to get moved around or covered in favour of those that are. When these older windows are needed by the user again, the windows on top get moved around or shuffled underneath the older windows, uncovering them for use.

  On a desktop, pieces of paper can be moved and shuffled around, so that urgent messages are at the top of the pile, and relatively unimportant ones drift to the bottom. Over time, importance changes, and these bottom messages will move back to the top of the stack.

- Being able to move and order windows on a desktop is not enough, unfortunately. As in our desktop metaphor, a desktop with a sufficient amount of paper on it becomes very disorganized and difficult to work with. In these cases, unneeded paper may be removed from the desktop and organized elsewhere, to return to the desktop when it is later needed.

  In a windowing environment, there is a corresponding notion of closing and reopening windows, more commonly referred to as minimizing and restoring windows. When a window is no longer needed, it can be removed from the desktop by closing or minimizing the window. When needed again, the window can be reopened or restored. While absent from the screen, an icon is usually left behind to remind the user that the window still exists and was not removed permanently. Consequently, the acts of removing and replacing windows on the screen is also referred to as iconifying or uniconifiying a window.

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

- The larger the desktop one has, the more material can be stored upon it. Not only does this apply to papers in our desktop metaphor, but also to windows in a windowing environment. Unfortunately, the size and resolution of the computer screen being used ultimately limits the space available for displaying windows in a windowing environment.

  To get around these limitations, many environments allow users to have virtual desktops or multiple desktops. Virtual desktops allow users to have window desktops whose size exceeds the physical limitations of their screens–the area currently displayed to the user can be panned around the virtual desktop so that the applications in use are visible, and those that are not are off-screen in regions of the virtual desktop not being displayed. Having multiple desktops to hold application windows would be similar to having multiple desks containing different pieces of paper on them; as work is done, and the focus of activity changes, the user switches between desktops as necessary so that the windows required are on the desktop currently being displayed on the screen.

- To secure windows from unauthorized access, particularly when a user is away from his or her computer, most windowing environments provide some kind of screen lock feature. This is similar to someone locking sensitive documents away within desk drawers when absent from their office, replacing them on their desktop when they return. Screen locks require users to enter a password to remove the lock and gain access to desktop windows. While locked, the screen is usually either blanked or displays some kind of graphical animation. This was done originally to prevent screen images from being permanently burned into the picture tube in early monitors; now, with improved monitor technology, these animations are primarily used for entertainment value.

  Similar to screen locks in this respect are screen savers. Screen savers were designed originally to "save" monitors from the burn damage described above, activating after the computer was left idle for some period of time. Now, however, screen savers are primarily used for entertainment purposes. Screen savers generally do not have a lock function built into them however, as they are meant to protect computer hardware rather than users' desktops and windows. To conserve energy, many of today's monitor manufacturers have screen savers in hardware – blanking the screen and reducing power usage after a certain idle period.

A great deal of information on how users use their computing environments can be obtained by observing and recording how they interact with their environments in the above fashions. Patterns of window usage are direct reflections of how users use their applications. Analysing and studying these patterns can shed light on how computing environments are used, which in turn can lead to the

Technical Report #540, Department of Computer Science, The University of Western Ontario,
London, Canada, November 1999

development of better environments. Several studies in the past have been carried out to this end [1, 2, 3], and have led to interesting results.

In this paper, we report on a similar study carried out in the Department of Computer Science at the University of Western Ontario. This initial study examines how users in this setting make use of their windowing environments by tracking the above kinds of interactions over extended periods of time. The results of this study have already been applied to the area of quality of service management, to optimize the scheduling and allocation of resources to user applications, to improve the overall level of quality of service being delivered to users [4]. Further applications of these results are currently underway.

This paper is structured as follows. Section 2 discusses the requirements of this study and outlines objectives to be met. Section 3 describes the particulars of how the study was designed and executed. Section 4 presents the results from the study, and formulated observations and conclusions based on these results. Section 5 concludes the paper with a summary of work to date, and a discussion of directions for this work to take in the future.

2 Requirements

The purpose of this study is to track user interactions with their windowing environments over time, to deduce individual and aggregate patterns for later use in environment improvements. To that end, the following requirements must be met:

- A reasonably large user population is required for statistical confidence in our results, and to obtain general patterns of windowing environment interaction. If the user population is large enough, it can be segmented into several different categories as well.

- Observations must span several days in order to provide sufficient data upon which statistics can be calculated and conclusions can be drawn. This will also provide additional confidence in our results by reducing the influence of unusual or abnormal activity over short periods of time.

- We must be able to collect data on the different types of interactions discussed in the previous section, in order to gain some basic ideas over how users work with their windowing environments. Ideally measurements should be taken continuously so that a complete set of observations is obtained. Doing so, however, may require recording a great deal of information. If the amount of information required becomes excessive, then some form of compression, filtering, or sampling may become necessary.

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

- The approach taken in the study should be applicable to a wide variety of windowing environments. In an initial study such as this, we will likely need to restrict ourselves to a single environment. However, the approach we take in collecting and processing data should be applicable to other environments to support further studies in the future.
- The process of collecting data on user interactions should be as unobtrusive as possible. We do not want to interfere with the way users interact with their windowing environments in participating in this study; this would only serve to distort our findings. As a result, the process of collecting data should be automatic and induce little load on the workstations and systems participating in the study.
- Finally, approval and support are needed from systems administrators for the study to take place. The data collection process may needs special access to collect the data it needs, and may need to alter the windowing environments in use. While both of these needs should be minimized in developing data collection tools, administrators should still be notified in case problems occur.

With the above requirements in mind, the next section of this paper details how the study was designed and carried out.

3 Design and Execution

Having given requirements for our study into user-windowing environment interactions in the previous section, this section examines how our study was designed, set-up, and executed.

3.1 Study Design

This study was designed to meet the requirements outlined in Section 2. Several design decisions were faced in preparing for this study; these decisions are outlined below.

3.1.1 Windowing Environment Selection

Because of differences between windowing environments, it was decided to restrict this study to examining the interactions of users in only one kind of environment. This way, we could determine the effectiveness of our approach and analysis before developing a large collection of tools to support multiple environments. Furthermore, nearly 75% of the systems used in our department make use of the same environment,

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

so limiting the study to a single environment does not substantially restrict the number of potential participants. Additional environments will be examined in the future.

For this study, the X Window System [5] was selected as the initial windowing environment to examine. This was done for several reasons:

- X is available on virtually every system in our department, whether they use Unix or Microsoft Windows. It is used on about 75% of those machines. As a result, we have a large user population to work with.

- The X protocol and client-server nature gives us the ability to collect data from user's environments remotely, making it easier to pool data together for later analysis. The X protocol also allows us to collect this data without modifying the windowing environment or applications in use.

- The X Window System is very well documented and is quite open. This facilitates the development of the data collection tools needed in the study.

- Data collection tools can be started automatically by simple modifications to users' X initialization files. This allows the data collection process to be unobtrusive to users' normal routines.

- No system administrator access is required for data collection under the X Window System. Once a user agrees to participate in the study, no further permission is required.

Since the X Window System met many of our requirements from the previous section, it was a natural choice for our initial study.

3.1.2 Data Selection

To meet the objective of this study, we must collect data that allows us to track user interactions with their windowing environments over time. However, tracking every user interaction is not feasible; the sheer volume of data would noticeably impact the operation of the user's windowing environment and overwhelm log storage capacity. So, instead of recording every mouse motion, mouse button click, and key press, we will only record those that indicate a significant interaction with the environment. These include:

- Iconifying and restoring of windows,

- Covering and uncovering of windows,

- Creating and destroying of windows,
- Screen saver and screen lock activations and deactivations,
- Changes in window focus (which windows are being interacted with).

These interactions were selected as they reflect a potential change in what the user is doing in his or her environment – very significant interactions. In future extensions, additional interactions may be collected as well, depending on the success of this initial study.

Data will be collected continuously over several one week periods, as time and disk space permits. Data collection will stop only if a user terminates a windowing environment session; collection will resume when a new session begins. Since only the above interactions are recorded, sampling is not necessary to reduce data volume and continuous recording is feasible.

3.1.3 Participant Selection

The last design decision is to determine criteria for study participants. Several conditions must be met for prospective participants to be suitable:

- They must use the X Window System on a regular basis. This is necessary to obtain sufficient meaningful data for statistical analyses.
- They must be willing to have their interactions, as described in the previous section, logged.
- They must have an account on the department's research network – this includes graduate students, faculty and staff. Although undergraduate students are also potential candidates for this study, their environments are under much greater strain and workload, which might complicate matters. Pending the success of the initial study, it will be extended to this population in the future.

To give participants sufficient time to volunteer, two weeks will be set aside for recruitment. At this point, any further instructions will be supplied to participants, and the study will begin.

3.2 Study Preparation and Setup

Since the study is reasonably straightforward, there is very little preparation needing to be done beforehand. All that needs to be done is the development and installation of tools to be used in the study, and the training of study participants in how to use them.

3.2.1 Tool Preparation and Setup

For data collection, a tool was developed called the *activity_recorder*. This tool connects to the X server on a participant's workstation or terminal to collect events indicative of the user interactions selected in Section 3.1.2. It does so by traversing the tree used by X to store information on all of the windows being displayed by the server; for each application window found, the tool subscribes for the necessary events. As application windows are created or deleted, the tool subscribes and unsubscribes for events as needed. A man page for the *activity_recorder* can be found in Appendix A.

When *activity_recorder* is first started, it records information on the user and the user's display. It then takes an inventory of all application windows on the display, and logs the window identifier, name, state, and originating client host for each window. As users interact with their windowing environment, the X server sends events to the subscribers of events for the windows affected by the interactions. This includes both the application owning the window, so it can handle the event appropriately, and the *activity_recorder*, so the event can be logged. Logged information includes the time of the event, the identifier of the window that experienced the event, and the type of event. The *activity_recorder* terminates only on explicit user request, or when the current X session terminates. A sample partial log generated by the *activity_recorder* can be found in Appendix B.

To store data logged by *activity_recorders* executed by participants in the study, a special storage area was created on our research network. A special shell script was used to execute each *activity_recorder* to ensure that log files were created in this area, and to ensure that logs, once created, were only readable or writable by *activity_recorder* processes and by the conductors of the study. This was to ensure the privacy of the information logged (although no sensitive data was actually being recorded), and to ensure the integrity of the collected data. In addition to these execution scripts, additional shell scripts were written to help filter and analyse collected data.

3.2.2 Participant Recruitment and Training

The last step in preparing for the study was to recruit participants. This was done to ensure that the study was completely ready to begin once we had enough participants; we did not want to keep people gracious enough to help us wait for any great length of time while other details were sorted out.

To recruit participants for our study, an e-mail message was sent out to all users of the department's research network. (A copy of this letter can be found in Appendix C.) Once the initial recruitment letter was sent, responses were collected for one week, after which a followup letter was sent to field any additional interested parties. The list of respondents was checked to ensure that all participants did, in fact, make use of X on a regular basis as their windowing environment. Acceptance letters were sent to all successful recruits.

After accepting participants, instructions were e-mailed to all of them explaining how to participate in the study. This included the following:

- How to start data collection manually.
- How to start data collection automatically.
- How to check that data collection is occuring.
- How to stop data collection.
- How to obtain additional help if necessary.

A copy of the complete set of instructions can be found in Appendix D. In addition to the e-mailed instructions, conductors of the study individually visited each participant to verify that things were proceeding appropriately, and to provide individualized help as necessary.

With this done, and all of the participants prepared, the study was ready to commence. Details of this are discussed in the next section.

3.3 Study Execution

A total of 12 participants were recruited for this initial study (6 faculty members, 3 staff members, and 3 graduate students). This was an ideal number for this initial study – enough participants for significant results, and not too many to overwhelm us while we first carried out the study. With the lessons learned from this initial study, we can plan for and execute a larger study in the future.

The study began in January 1999, and ended in February. Seven weeks of data were collected on each participant – 20 megabytes of data in total. There was found to be enough data for analysis of individual patterns, as well as aggregate patterns over all participants. The results from this analysis are discussed in the next section.

4 Study Results

In this section, we examine some of the results obtained from this initial study. These results are summarized in the sections below.

4.1 User Access Patterns

Each user that participated in the study accessed their windowing environment every weekday throughout the study. Access on Saturdays and Sundays was sporadic, and only a few users accessed their environments on those days, if at all.

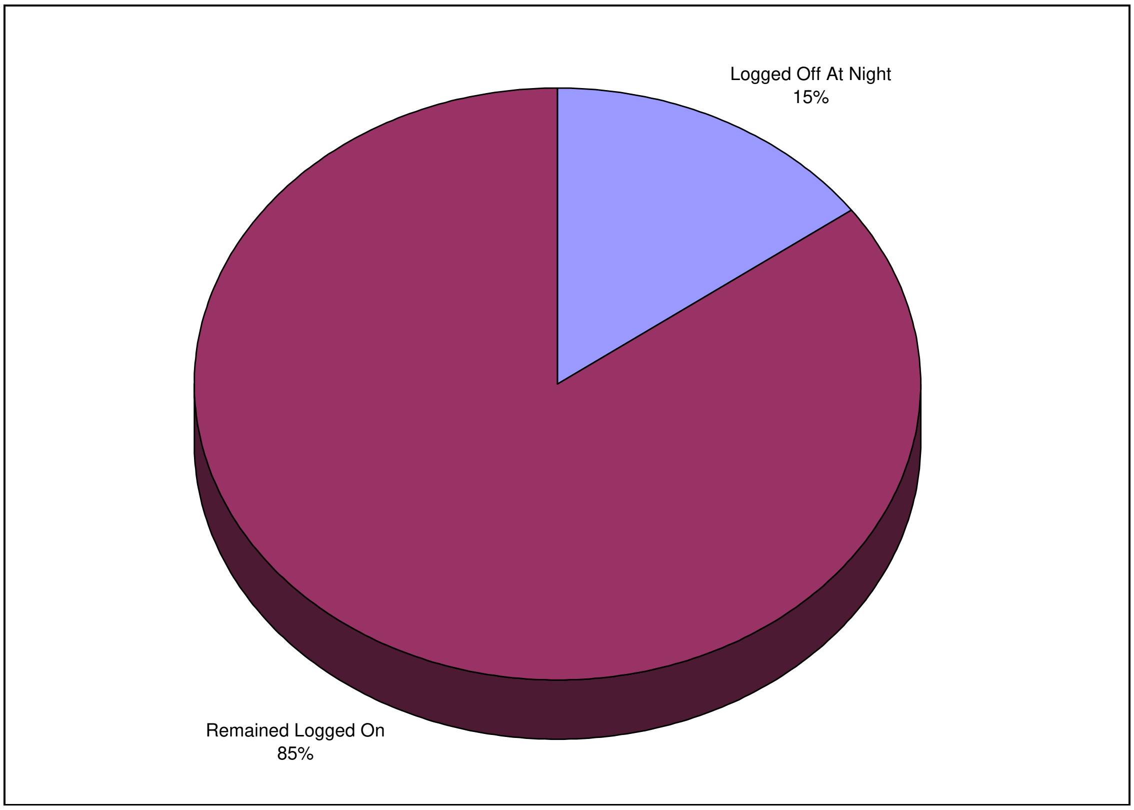

Figure 1: Nightly Environment Access

Since each participant had either a private or semi-private office, with their a workstation allocated for their own personal use alone, interesting effects were noticed. As seen in Figure 1, only 15% of participants logged off at night, ending their sessions. The remaining 85% remained logged on, leaving their windowing environment and applications executing over night. This was largely due to the fact that the users did not need to share their workstations with other users, allowing the users to break off a session at night and continue it in the morning without having to log back in and restart their applications.

4.2 User Activity Patterns

After each participant's activity over time was plotted, it became clear that there were distinct patterns to this activity. Furthermore, it was seen that each participant's activity patterns differed from that of the other participants. This would suggest that there is some predictable way in which the participants behaved – arriving and leaving at the same times, taking lunch at the same times, and so on.

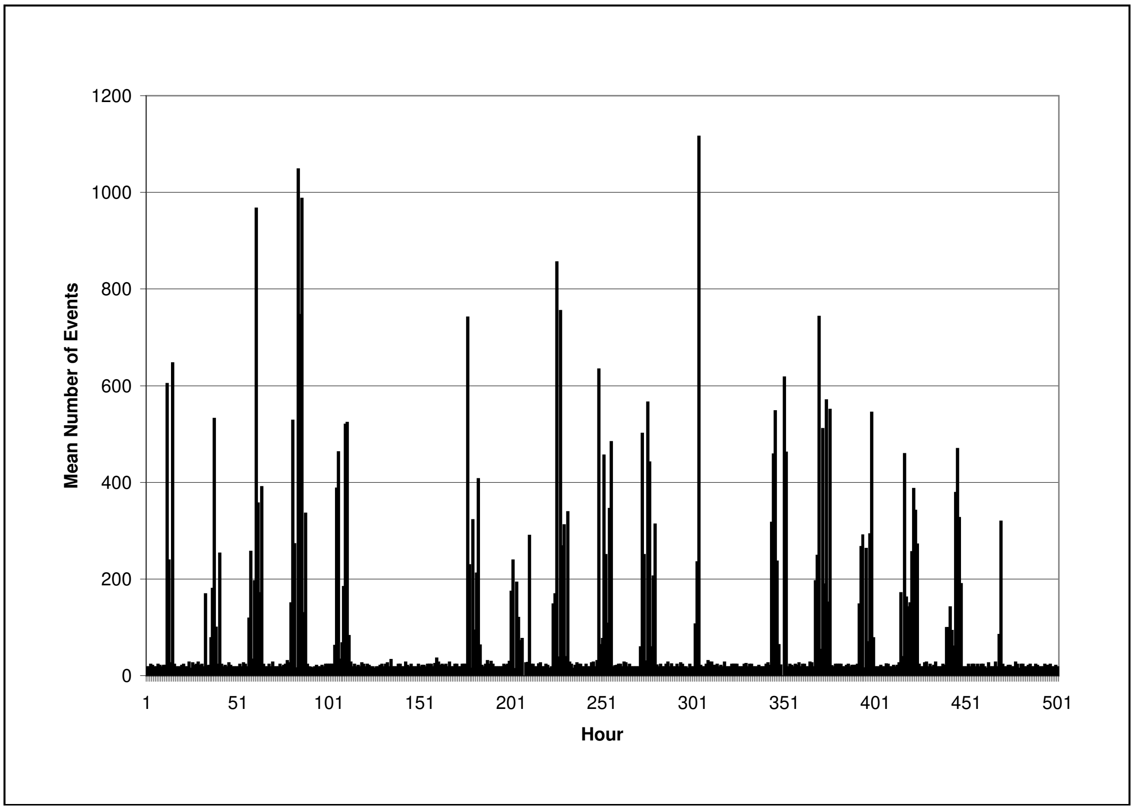

Figure 2: One Participant's Activity over a Three Week Period

For example, Figure 2, tracks the activity of one participant over a three week period. Activity is gauged in terms of the mean number of windowing environment events capturing the various interactions of Section 3.1.2 measured per hour. As can be seen, there were definite patterns to this user's activity. Activity was regular each week day, and rare on Saturdays. Activity was minimal on Sunday and at nights, with a base line of activity likely caused by the screen saver or locking mechanism in place.

Figures 3 and 4 summarize the activity from Figure 2 over periods of a week and a day, respectively. The summaries were constructed by plotting the mean of the raw data plotted in Figure 2; low deviations from doing so indicate that the summaries are reasonably accurate reflections. These summaries were also validated through checking with participant's other activities. For example, the participant whose

Technical Report #540, Department of Computer Science, The University of Western Ontario,
London, Canada, November 1999

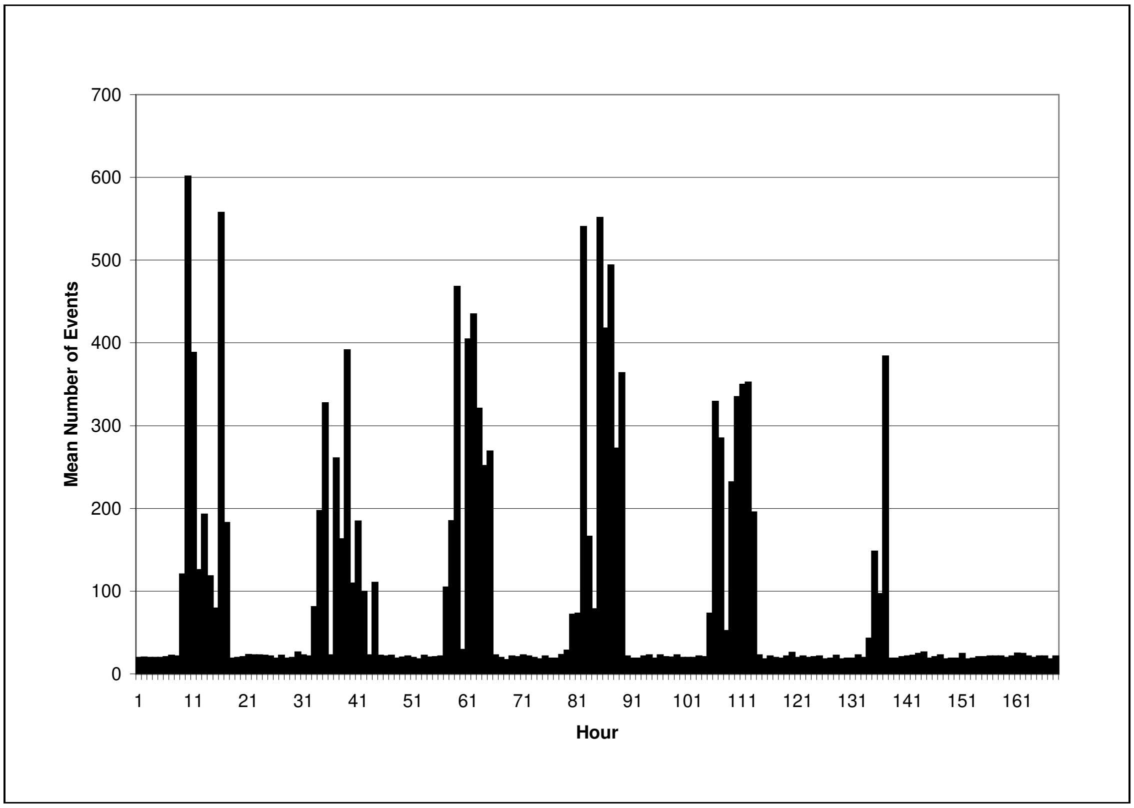

Figure 3: One Participant's Average Activity During a Week

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

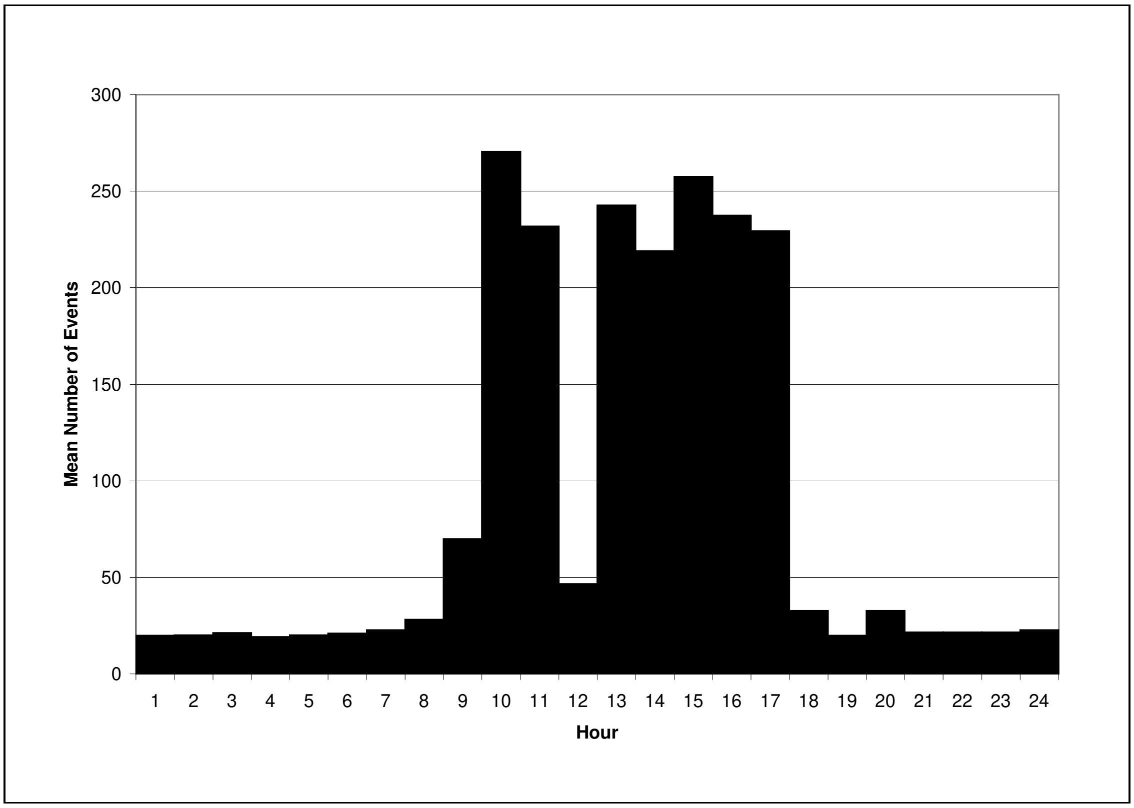

Figure 4: One Participant's Average Activity During a Day

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

activity was plotted in Figures 2, 3, and 4 arrived for work between 9:30 and 10:00 in the morning, usually left before 6:00 in the evening and played cards during lunch from 12:00 to 1:00. This correlates closely with the pattern from Figure 4.

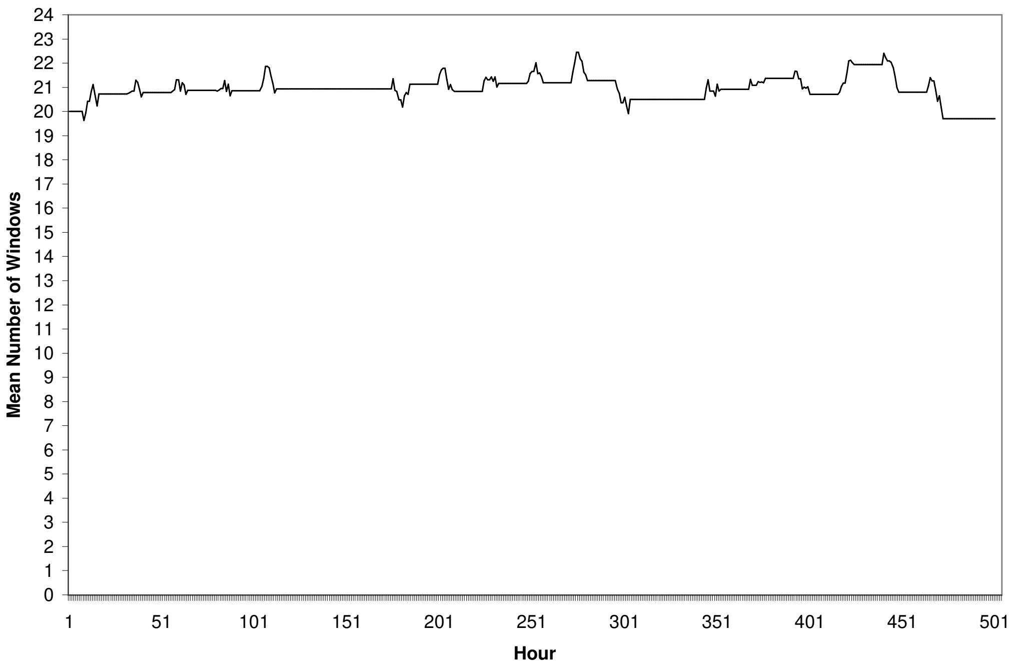

Figure 5: One Participant's Window Usage Over Three Weeks

It is interesting to note, however, that despite the range in activity of every participant, each participant always maintained a minimum number of windows on his or her desktop. (The only exception, of course, were those people who logged off at night, as noted in the previous section.) For example, Figure 5 shows the mean number of windows open during the same three week period for the same participant. While there were changes in the number of windows in use during user activity, there was always at least 19 windows open at all times, even at nights and on weekends when the only activity on the workstation was due to the screen saver or lock in place.

4.3 User Window Usage

In this section, we examine a variety of analyses dealing with how study participants made use of windows in their environment. The results of these analyses were quite interesting.

Technical Report #540, Department of Computer Science, The University of Western Ontario,
London, Canada, November 1999

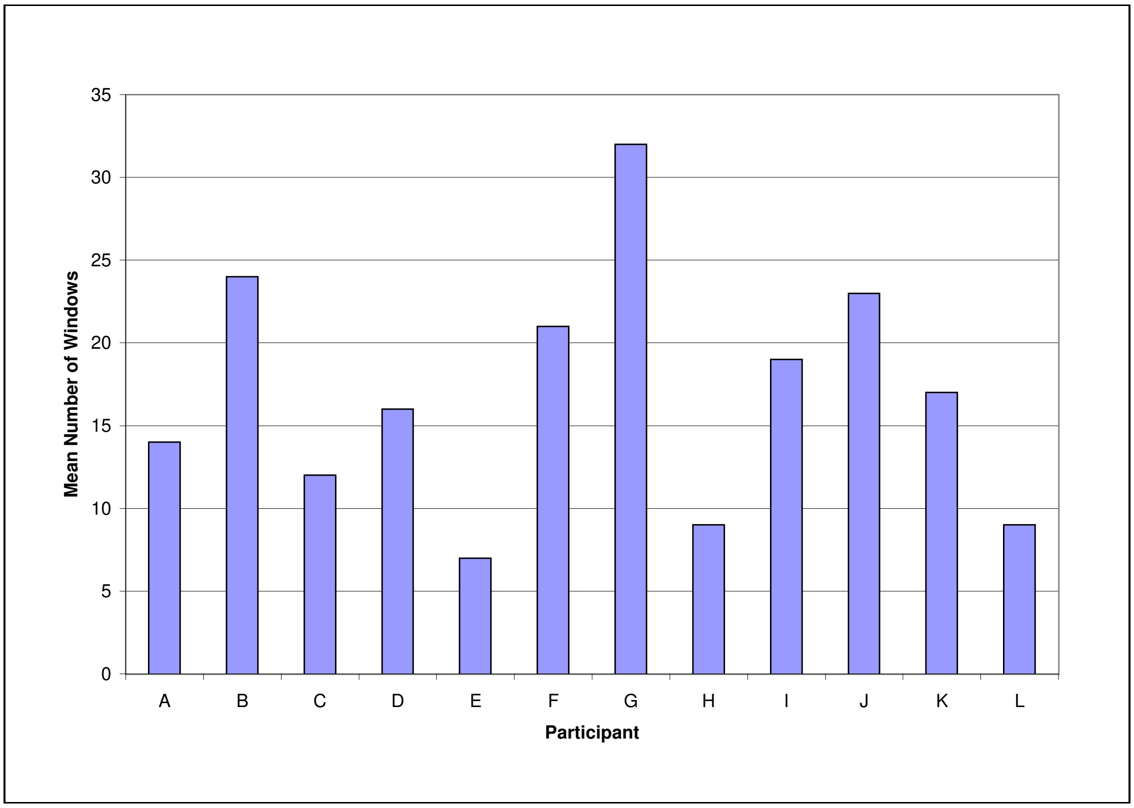

Figure 6: Mean Number of Open Windows for all Participants

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

An analysis of the mean number of open windows on participant's desktops produces several results, as can be seen from Figure 6. First, it can be seen that different users make use of their environments differently. Some users had only a few windows open on average, while some had dozens open. This reflects their personal preferences, but also likely the capacity of their workstation and the window manager in place. Second, it was found that on average, users had nearly 17 application windows open at any given time. This is a fairly substantial amount, indicating that the users are indeed taking advantage of their environments. Third, it was found that less than 15% of the study group had less than 10 windows open on average. Again, this indicates the high usage of these environments.

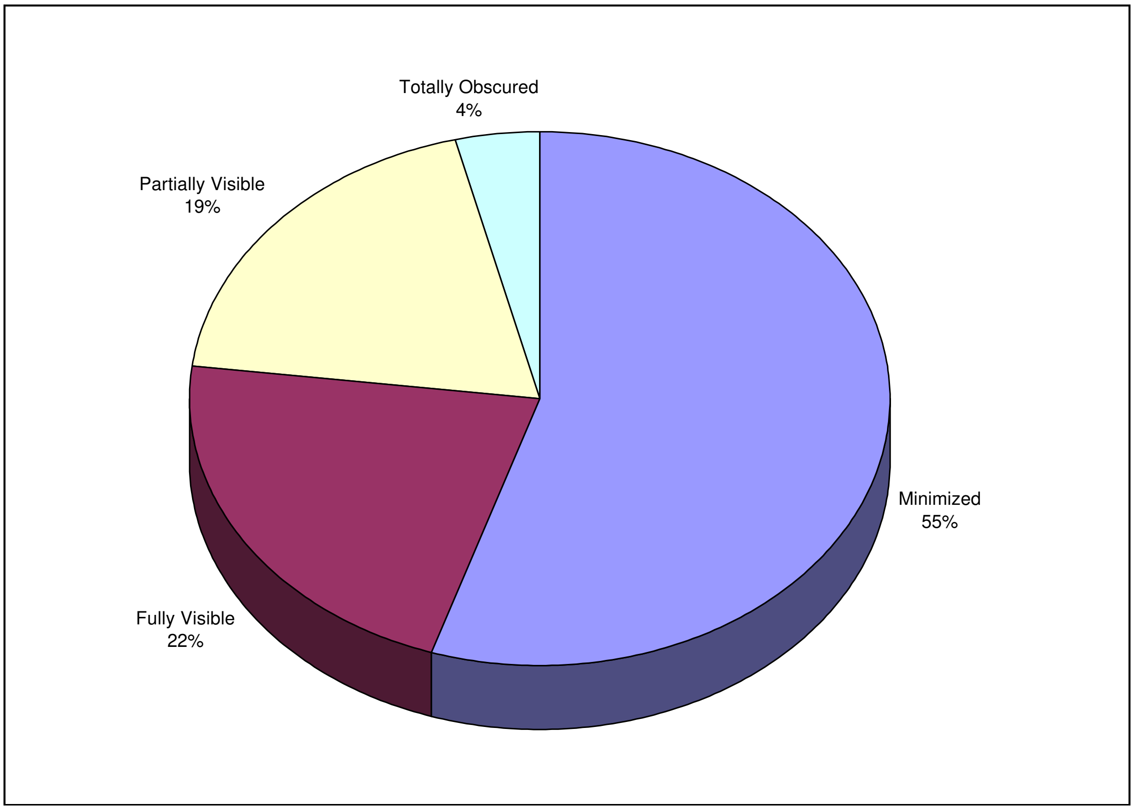

Figure 7: Percentages of Windows in Different States

Understandably, one cannot make use of 17 application windows all at once on top of a windowing environment desktop. Some windows will need to be covered up, and others minimized. An analysis of user activity causing changes in window state provides a view of what percentage of application windows are in each state at any given time, as shown in Figure 7. On average, only 22% of applications executing were completely visible and usable to the user. Nearly 55% of the applications were minimized, 19% were partially visible, and 4% were totally obscured. This indicates again that the users are making effective

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

use of the capabilities of their windowing environments.

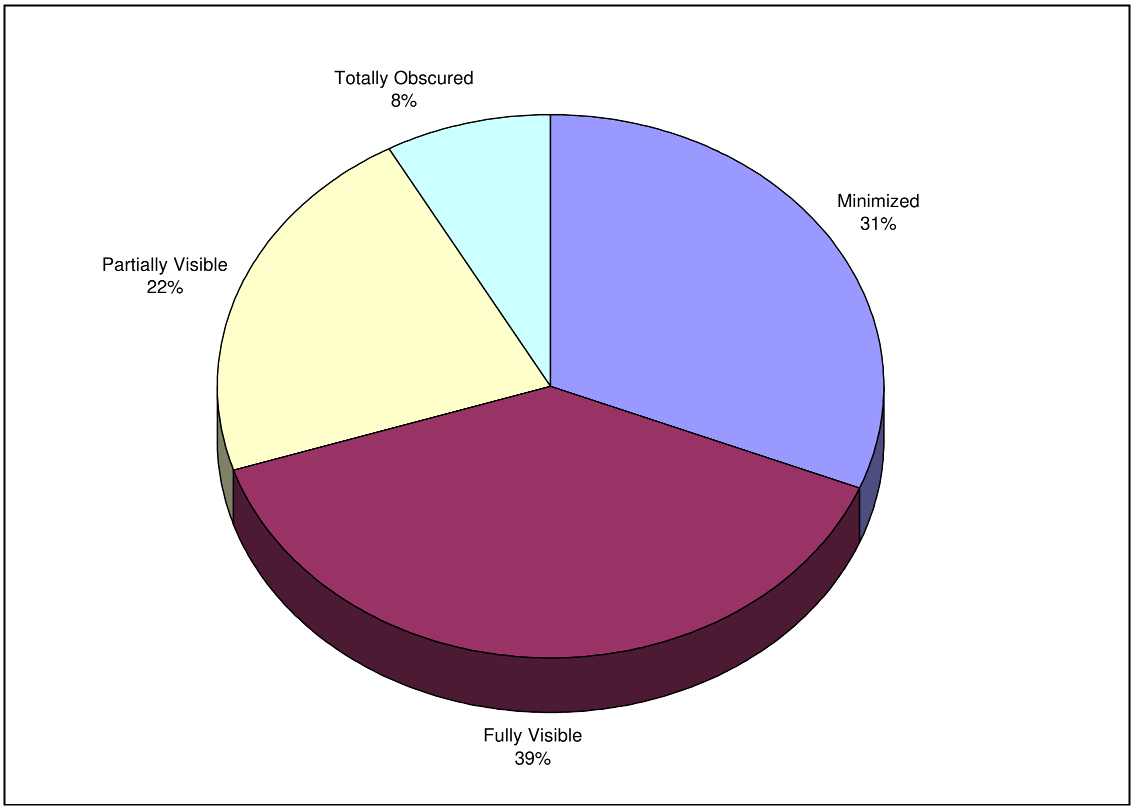

Figure 8: Percentages of Window Life Time in Different States

In the end, as shown in Figure 8, the average application spends only around 40% of its life completely visible. The rest of its life is spent either minimized or covered by other windows in some way. This, naturally, depends on the windows in use. Some windows, such as dialog boxes, had life times of only seconds during which they were always completely visible; others had life times of several weeks during which the windows spent most of their time covered or minimized. In general, it was found that the shorter the life time of the window, the larger the percentage of life time was in a completely visible state. This reflects results from [1, 2, 3] that showed that users rarely complete any lengthy task before beginning another, resulting in windows from long-running tasks to be obscured or rendered otherwise unusable while other tasks were being processed.

By examining changes in window focus, we were able to see how often users changed tasks in their environments. When their workstations were in use and no screen savers or locks were present, as discussed in the next section, we measured that users could change focus up to 50 to 100 times or more per hour in peak usage. 18 times per hour was the overall average, when the user was working at his or

Technical Report #540, Department of Computer Science, The University of Western Ontario,
London, Canada, November 1999

her workstation. This kind of activity is once again supported by the studies carried out in [1, 2, 3].

4.4 User Screen Saver and Lock Usage

Many study participants did not use a separate screen saver, but rather relied on the built-in screen blanker – either in their video hardware, or in their X server software. 42% of participants used a separate screen saver, primarily for entertainment purposes. Of the users that remained logged in at night, it was nearly a 50-50 split as to whether the users employed a screen lock or just a screen saver.

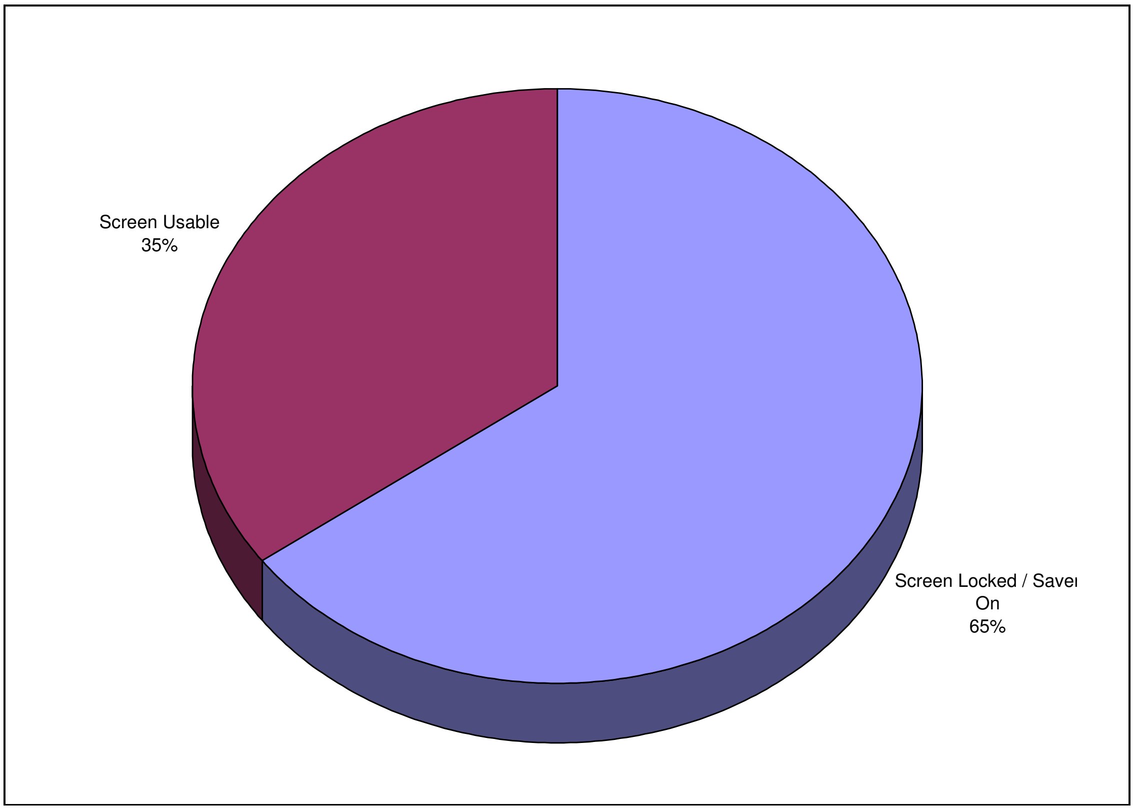

Figure 9: Screen Saver and Lock Usage Time

Figure 9 shows the percentage of time that the average workstation spends with either a screen saver or lock activated (providing that the user does not log off at night). Nearly 65% of the time, a saver or lock is operating – the user is either away from his or her computer, or has gone home for the night or weekend. Regardless, the user's applications are still executing underneath the lock or saver with windows open, as shown in previous results.

5 Conclusions and Future Directions

In this paper, we presented an initial study that tracked user interactions with their windowing environments. From this, we learned or confirmed several interesting things, including:

- Users display distinct patterns when accessing their windowing environments.
- Users with private workstations tend not to terminate sessions, but rather leave themselves logged in 24 hours a day, 7 days a week.
- Users take advantage of the capabilities of their environments to work with and manipulate large quantities of application tasks and windows.
- Users change the task they are working with rather frequently, and seldom finish a long-running task before commencing a new one.
- Each user often has many application tasks that are in unusable states at any given time.
- During the average day, a private workstation tends to spend more of its time locked or with a screen saver activated than it spends open with its user working on it.

It is possible to use the results of this study to help design and build better environments for users to work in. As mentioned previously, the results of this study have already been applied to the area of quality of service management, to optimize the scheduling and allocation of resources to user applications, to improve the overall level of quality of service being delivered to users [4].

We have identified several avenues for continuing and future work in this area:

- Extend the study to include more participants.
- Extend the study over even longer periods of time.
- Support additional platforms, most notably Windows $9x$/NT.
- Conduct more complex analyses on the data collected to deduce additional trends and patterns.
- Research further applications of the study's results.

Based on the success of this initial study, plans for further investigations incorporating some of the above extensions are in the works for the coming months.

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

Appendix A – activity_recorder Man Page

```
User Commands                                ACTIVITY_RECORDER(1)



NAME
     activity_recorder - program for collecting and recording user
     activty in X Windows

SYNOPSIS
     activity_recorder [options]

DESCRIPTION
     activity_recorder is used for collecting user activity
     in an X Window session.  These include changes in focus,
     covering and uncovering windows, minimizing and restoring
     windows, and screen saver and screen lock activation and
     deactivation.  By collecting and logging this activity, we can
     determine how users use their windowing environment and how
     we can improve the environments as a result.

     Any errors encountered in the process will be reported
     immediately, and cause the program to abort.

OPTIONS
     -f file-name
          The name of the file to log activity to.  Setting this
          file to - will log activity to standard output.
          (Defaults to standard output.)

     -d display
```

The name of the X display to log activity from. If this parameter is not given, the DISPLAY environment variable will be read and used instead.

-t This option forces the program to log times as absolute times as opposed to the number of seconds that have elapsed since the program started executing.

-r This option forces the program to raise all mapped (not iconified) viewable windows when it starts. When done this procedure, the desktop will be in exactly the same state as before the program started. However, doing this allows the program to get visibility events to determine which windows are completely visible, partially visible, or totally obscured. Without these events, the program could not tell how windows are covered when it first starts.

-s This option instructs the program to also monitor the built-in X screen saver that is part of the X server. Third-party screen savers, like xscreensaver map an additional graphics hack window to do animations and keep the screen from burning in. This mapped window completely obscures all other windows, allowing us to detect when it becomes activated. However, the built-in screen saver only blanks the display, and generates no events of its activation (unless the MIT X Screen Saver extension is supported by both the client and server systems, which is not likely). As a result, additional monitoring must be done to determine when the screen saver is activated and deactivated. Since this monitoring may induce additional server load, this monitoring is optional.

-h This option causes the program to print a simple help message and then exit.

EXAMPLES

To capture a default activity log to standard output:

```
activity_recorder
```

To capture an activty log, with windows raised to determine inital visibility, to the file foo.log:

```
activity_recorder -r -f foo.log
```

To capture an activity log, with windows raised to determine initial visibility and screen saver activity also logged, to standard output:

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

```
activity_recorder -r -s
```

To capture a default activity log with absolute time timestamps instead of elapsed time:

```
activity_recorder -t
```

DIAGNOSTICS

The exit status of the program is 0 if and only if all operations were successful. Otherwise, an exit status of 1 is produced.

SEE ALSO

xev (1), xwininfo (1), xset (1), xscreensaver (1)

COPYRIGHT

AUTHOR

Michael J. Katchabaw, of the Department of Computer Science, the University of Western Ontario, London, Ontario, Canada.

BUGS

Please send bug reports to katchab@csd.uwo.ca.

NOTES

There are two additional classes of user activity that would be REALLY nice to be able to log. The first involves power management. Most modern systems (and some archaic ones) provide hardware support for sending monitors and other devices into a low power mode after a certain amount of user inactivity. In these cases, screen savers (built-in and otherwise) may not be used as the monitor blanks the screen in hardware itself when necessary. This power management is another hint of user activity, as they are not interested in their display if these features kick in for extended period of time. Unfortunately, the various platforms handle this in system specific and hardware specific ways, making the detection of these hints practically impossible.

The other activity that would be useful to detect involves the paging of displays with window managers, like FVWM and others, which support multiple desktops and multiple virtual pages on a desktop (or a virtual desktop that exceeds the physical display). Just as when applications are covered or minimized, applications that are not on the current desktop, or are off of the physical screen of a large virtual desktop, are not the focus of user activity. As a result desktop switching, movement, and other such activity are things that are of great interest. Unfortunately, different window managers handle these actions in different ways, often in ways that are incredibly difficult to detect. If they can be detected, it is often difficult to determine what exactly is happening based on the limited amount of information available from the X server or the window manager in use.

Taking FVWM as an example, if the page on the same virtual desktop is switched windows will be obscured and unobscured, which can be detected along with normal hints. However, if you switch to the same page on a different desktop, all application windows will echo their current state in visibility events. This can be used to determine when the page change occurs, but one cannot tell from this alone which applications are on the current desktop and which ones are not. (One could try and check the viewability state of the application, which would indicate that the window's parent is unmapped, but this information is not available when the visibilty events occur.) Sigh.

As a result, both of these classes of hints are the subject of continuing work on this application.

Appendix B – Partial activity_recorder Log File

```
Logging started at Mon Jan 11 18:03:23 1999
0.001565 :  activity_recorder.bin, Version 1.0
0.002030 :  Log generated for user katchab (10387) on display xcel:0.0
0.002091 :  Window ID 0x23 is the root window (1280x1024)
0.088568 :  Window ID 0x2c00020 has name "TWM Icon Manager" ("None" "None"),
            state IsUnmapped, from client unspecified
0.130706 :  Window ID 0x1c0000d has name "xterm2" ("xterm2" "XTerm"),
            state IsViewable, from client brown.csd.uwo.ca
0.151464 :  Window ID 0x2800007 has name "oclock" ("oclock" "Clock"),
            state IsViewable, from client brown.csd.uwo.ca
0.260474 :  Window ID 0x200000d has name "Brown" ("Brown" "XTerm"),
            state IsViewable, from client brown.csd.uwo.ca
0.307226 :  Window ID 0x300003e has name "pleaseStandBy" ("pleaseStandBy" "Xman"),
            state IsUnmapped, from client brown.csd.uwo.ca
0.346846 :  Window ID 0x3000029 has name "Manual Page" ("manualBrowser" "Xman"),
            state IsUnmapped, from client brown.csd.uwo.ca
0.405534 :  Window ID 0x180000d has name "xterm1" ("xterm1" "XTerm"),
            state IsViewable, from client brown.csd.uwo.ca
0.473318 :  Window ID 0x240000d has name "xterm3" ("xterm3" "XTerm"),
            state IsViewable, from client brown.csd.uwo.ca
0.473556 :  Window ID 0x1c0000d event unobscured
0.473606 :  Window ID 0x2800007 event partially-obscured
0.473652 :  Window ID 0x2800007 event unobscured
0.500607 :  Window ID 0x200000d event unobscured
0.544709 :  Window ID 0x180000d event unobscured
0.544794 :  Window ID 0x200000d event partially-obscured
0.544838 :  Window ID 0x1c0000d event partially-obscured
0.579625 :  Window ID 0x240000d event unobscured
0.579694 :  Window ID 0x180000d event partially-obscured
0.610783 :  X screen saver is enabled
```


Technical Report #540, Department of Computer Science, The University of Western Ontario,
London, Canada, November 1999

```
114.510202 :  Window ID 0x180000d event unobscured
114.510300 :  Window ID 0x240000d event partially-obscured
124.063874 :  Window ID 0x180000d event partially-obscured
125.976527 :  Window ID 0x180000d event unobscured
126.300984 :  Window ID 0x4800008 has name "emacs@brown.csd.uwo.ca" ("emacs" "Emacs"),
              state IsViewable, from client unspecified
126.301091 :  Window ID 0x2c00020 event configured
126.301145 :  Window ID 0x180000d event partially-obscured
128.398730 :  Window ID 0x4800008 event configured
128.398878 :  Window ID 0x4800008 event partially-obscured
128.445617 :  Window ID 0x4800008 event unobscured
815.803334 :  Window ID 0x240000d event unobscured
815.803433 :  Window ID 0x4800008 event partially-obscured
830.105579 :  Window ID 0x4c0023c has name "Netscape Navigator 3.01" ("Navigator"
              "Netscape"), state IsViewable, from client brown.csd.uwo.ca
830.105700 :  Window ID 0x2c00020 event configured
830.105754 :  Window ID 0x240000d event partially-obscured
830.105800 :  Window ID 0x2800007 event partially-obscured
...
```

Appendix C – Participant Recruitment Letter

Hello all,

As part of some of our on-going research in the Department of Computer Science here at Western on quality of service management and provisioning, we are about to commence a study, and we would like your assistance. Participating in the study is quite simple, so please read on ...

Basically, we are interested in how people interact with and make use of their windowing systems and desktops. This information can then be used to help us provide better quality of service to these users. For this study, we are focussing on people using the X Window system on the variety of Unix hardware in the department (including X terminals, Sun workstations, AIX workstations, Linux workstations, and so on), as well as people who are using X Windows through programs like XWin32 from their Windows NT workstations.

The information that we will be collecting consists of a variety of events triggered by your interactions with the windowing system, including:

- Iconifying and restoring of windows.
- Covering and uncovering of windows.
- Creating and destroying of windows.
- Screen saver and screen lock activations and deactivations.
- Changes in window focus (which windows you are interacting with).

We will not record any additional information besides the events listed above, for obvious privacy reasons. The information we do collect will naturally be confidential (not that there is much use to it, besides our analysis!!!).

To participate in this study, you don't actually need to do anything yourself, except for executing one of our management applications that records the above events for us when you start an X Windows session. (If you add this application to your .xsession or .xinitrc file, you don't even need to do that!) You can stop the recording whenever you want, or leave it up to the application to do so when you shut down your X Windows session. It's that simple. We'll take care of the rest.

If you are interested in participating in this study or have any questions, please contact Mike Katchabaw at katchab@csd.uwo.ca.

Thank you for your time and attention!

Mike.

Appendix D – Instructions for Participants

Hello,

Thank you for participating in the windowing system usage study! This message contains information and instructions on how to use the program to collect the necessary information for this study. Please read on, and if you have any questions at all, direct them to Mike Katchabaw at katchab@csd.uwo.ca.

Getting Started

To participate in this study, you only need to execute one of our simple management applications to collect events from your X Server. There are two ways to do this:

1) Start X Windows as you ordinarily do. From an xterm, execute the following command:

```
/csd/grad/katchab/activity-recorder/activity_recorder
```

   It is important to note that this must be done on the research network from a machine using the Solaris operating system! The best machine to do this from is brown, which shouldn't be a problem for most people as this tends to be the default server most people use (it is also the default server for most, if not all, X terminals in the department). To find out if the machine you are logged in to supports the program, simply execute it, and the following message will appear if the program is not supported:

```
activity_recorder:  Sorry, can only be executed on Solaris 2.5 or above.

                Please log into a machine such as brown, set the DISPLAY,
                variable and try again.  Contact katchab@csd.uwo.ca for
                further details and instructions.

If this happens, the easiest thing to do is log in to brown, set your
DISPLAY environment variable, and execute the program.  If you have
created a .rhosts file that allows you to log into other machines
with out your password, the easiest way to execute the program is
by executing either:

  xon brown /csd/grad/katchab/activity-recorder/activity_recorder

or alternatively,

  xrsh brown /csd/grad/katchab/activity-recorder/activity_recorder

If you don't have a .rhosts file, you can do things easily yourself,
by following the steps below.

1.  Allow brown to connect to your X display by executing "xhost +brown".

2.  Find the current DISPLAY setting for your xterm by executing:
    "echo $DISPLAY".

3.  Login to brown by executing "rlogin brown.csd.uwo.ca".

4.  Set your DISPLAY environment variable to the value resulting
    from Step 2.  If you are using csh or tcsh as your shell, this
    can be done by executing "setenv DISPLAY x", where x is the
    display name returned from Step 2.  If you are using sh or some
```


Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

other Bourne shell variant as your shell, this can be done by executing "DISPLAY=x" and then executing "export DISPLAY", where x is the display name returned from Step 2.

5. Now execute "/csd/grad/katchab/activity-recorder/activity_recorder".

6. If you do not want to be logged in to brown, exit the connection, and return to work.

7. If you do not want brown to connect to your X server any more, execute "xhost -brown".

2) Instead of starting X Windows, and then the program, you can add the program to your .xinitrc or .xsession file so that it will be started automatically when you start your X Windows session. You will need to add the following line to your file BEFORE the window manager is started (which is usually at the bottom of an .xinitrc or .xsession file):

```
/csd/grad/katchab/activity-recorder/activity_recorder &
```

Once again, you should logged into a Solaris machine on the research network, such as brown, for this to work correctly. (Most X terminals will log you into brown by default, so this will work just fine too.) To see if the machine you routinely log into supports this, start up X as you normally would, and see the first approach above.

If the machine you log into routinely will not execute the program itself, you can still start it from your .xinitrc or .xsession files if you have a .rhosts file that allows you to rlogin to brown with out providing your password. If this is the case for you, you can then add either:

```
  xon brown /csd/grad/katchab/activity-recorder/activity_recorder &
```

or alternatively,

```
  xrsh brown /csd/grad/katchab/activity-recorder/activity_recorder &
```

If you don't have a .rhosts file, you will either need to create one (which I can assist you with), or you will need to start up X Windows first, and follow the first approach given above.

NOTE: When starting the collection of events, you might notice your windows reshuffled. If so, don't worry -- the program is just looking around to see what you have and where you have it!

Checking Things Out

If no errors are reported to you, then the program didn't have any problems in starting up. If you would like to verify that the program did start, doing so is quite simple. To make this check, simply execute the following:

```
/csd/grad/katchab/activity-recorder/activity_recorder_active
```

This must be executed on the same host that the activity_recorder program was executed on in the first place. So, if you used xon or xrsh to start the program, you can do the following to check its status:

```
xon brown /csd/grad/katchab/activity-recorder/activity_recorder_active
```

or alternatively,

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999

```
  xrsh brown /csd/grad/katchab/activity-recorder/activity_recorder_active
```

If you used rlogin to get to brown to execute the program, you need
to follow the steps below.

1. Login to brown by executing "rlogin brown.csd.uwo.ca".

2. Now execute "/csd/grad/katchab/activity-recorder/activity_recorder_active".

3. If you do not want to be logged in to brown, exit the connection,
   and return to work.

Depending on the message printed by the program, you can tell if the
program is running or not.

Stopping the Program

The program will stop collecting events automatically when your X session
terminates. There should be no need for you to stop the collection of
events, even when you go home for the night (that data is still important
too!). However, you can still terminate the logging yourself at any
time, if you so choose, by executing the following program:

```
  /csd/grad/katchab/activity-recorder/activity_recorder_shutdown
```

This must be executed on the same host that the activity_recorder program
was executed on in the first place. So, if you used xon or xrsh to
start the program, you can do the following to shut it down:

Technical Report #540, Department of Computer Science, The University of Western Ontario,
London, Canada, November 1999

```
  xon brown /csd/grad/katchab/activity-recorder/activity_recorder_shutdown
```

or alternatively,

```
  xrsh brown /csd/grad/katchab/activity-recorder/activity_recorder_shutdown
```

If you used rlogin to get to brown to execute the program, you need
to follow the steps below.

1. Login to brown by executing "rlogin brown.csd.uwo.ca".

2. Now execute "/csd/grad/katchab/activity-recorder/activity_recorder_shutdown".

3. If you do not want to be logged in to brown, exit the connection,
   and return to work.

You can then re-run the program at a later date using the methods described
above.

Additional Assistance

If you run into any difficulties or having any questions at all, don't
hesitate to contact me at katchab@csd.uwo.ca. If you like, I can drop
by your office and help you get things started, at a time of your
choosing. (I'll probably drop by and see how you are getting along
with it anyways!)

Once again, thank you for you assistance and participation!

Mike.

References

[1] L. Bannon, A. Cypher, S. Greenspan, and M. Monty. Evaluation and Analysis of Users' Activity Organization. *Proceedings of the 1983 ACM Conference on Human Factors in Computing Systems*, Boston, Massachusetts, December 1983.

[2] K. Gaylin. How are Windows Used? Some Notes on Creating an Empirically-Based Windowing Benchmark Task. *Proceedings of the 1986 ACM Conference on Human Factors in Computing Systems*, Boston, Massachusetts, April 1986.

[3] E. Kandogan and B. Shneiderman. Elastic Windows: Evaluation of Multi-Window Operations. *Proceedings of the 1997 ACM Conference on Human Factors in Computing Systems*, Atlanta, Georgia, March 1997.

[4] M. J. Katchabaw, H. L. Lutfiyya, and M. A. Bauer. Using User Hints to Guide Resource Management for Quality of Service. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, Nevada, June 1999.

[5] L. Reiss and J. Radin. *X Window Inside and Out*. McGraw Hill Incorporated, 1992.

Technical Report #540, Department of Computer Science, The University of Western Ontario, London, Canada, November 1999