Frequency-based detection of task switches

Rahul Nair
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA, USA
rahulnair@acm.org

Stephen Voida, Elizabeth D. Mynatt
GVU Center
College of Computing
Georgia Institute of Technology
{svoida, mynatt}@cc.gatech.edu

There is a growing amount of interest in learning about users' task patterns and task switches in daily life. This paper presents a novel method of detecting task switches based on the frequency of interaction with windows on a desktop computer. The results of a two week user study of its accuracy are also presented, along with possible applications for this type of system.

Modelling and prediction of user behaviour, task switching, window management, interaction frequency, user interaction

1. INTRODUCTION

Current HCI research is moving away from the traditional application–centric computing model towards a more activity–based approach. These new research initiatives have several different goals such as interruption recovery, determining interruptability, intelligent agents, and task archival. The common thread among much of this research is the need to accurately determine what task a user is performing, and by extension, determine when each task begins and ends. Current techniques used for this purpose include the use of external sensors (e.g., [3, 6]), application usage (e.g., [6]), and explicit user input (e.g., [7]). However, Czerwinski et al. [1] found that 69% of user task switches were either self–initiated (40%) or the result of moving on to the next task on a task list (19%). Since there are no external indicators when one of these switches occurs, these two categories of task switches are difficult to detect using external sensors. While user input could improve accuracy, requiring users to explicitly indicate changes in their tasks is intrusive and, since it is often tangential to their work practices, easily forgotten.

We present a new technique that uses the frequency of user interaction with desktop windows to infer when it is most likely that a task switch is occurring. We describe a user study we designed to test the validity of this approach and present its results. Finally we discuss the advantages and disadvantages of our approach as well as its possible applications.

2. INFERRING TASK SWITCHES USING WINDOW MANIPULATION FREQUENCY

There is a significant body of research that shows that users are very particular about the layout of windows on their desktop. As part of their work on the Rooms virtual workspace [5], Henderson and Card suggested that users have a "working set" of windows active at any given time. By moving between the windows of the tools (applications), users are implicitly defining this working set of windows. Studies [8, 9] of the window management behaviour of computer users found distinct differences in visible windows when users switched between tasks when compared to window switching during a single task. Users are as interested in hiding windows irrelevant to their current task as they are in displaying (and working with) relevant ones. Within–task switching was also found to involve fewer window manipulations than a switch between major tasks.

When a user switches between tasks, he/she opens the necessary task windows to his/her favourite positions — this is not restricted to opening new windows and can occur even if the user is merely rearranging pre–existing windows into a new configuration. We hypothesize that this activity produces window operations at a different frequency when compared to normal work. We use low-level observations of windows manipulations and their frequency distribution to determine when a task switch might be taking place. We believe that this technique can provide excellent indications of task switching without requiring explicit user input.

3. SYSTEM DESIGN

We have designed a background application that uses window manipulation frequencies to determine when a task switch occurs. The program tracks window events such as *create*, *activate*, *maximize*, *minimize*, *hide*, *show* and *destroy* by programmatically hooking into window system events that are publicly available on the Microsoft Windows operating system. This event information is time–stamped and augmented with details of the id, name, caption and class of the generating window before being sent to the detection algorithm and added to a log file.

3.1 Detection Algorithm

Our detection algorithm begins by filtering the window events so that only the events of top–level windows are used. The timestamps are then used to calculate the average time between window events for a given task. The system also calculates the simple moving average of the time between the last k interactions. When the ratio between the overall task average and the moving average exceeds a certain threshold, the system generates a new task event and resets the averages.

Example: Consider a situation where the algorithm is evaluating the n+1th window event. t_i *is the time between the ith and (i-1)th event:*

$$\text{Task average, } v_{TA} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

$$\text{Moving average, } v_{MA} = \frac{1}{k}\sum_{i=n-k}^{n} t_i$$

$$\text{Ratio} = \frac{v_{MA}}{v_{TA}}$$

Empirical testing has shown that a moving average of the last 7 windows (k = 7), coupled with threshold values of 0.67 (lower limit) and 1.5 (upper limit) provides a good compromise between accurate task detection and avoiding false positives. Further improvements may be possible by using an exponential weighted average or by dynamically altering the threshold and k values based on past performance.

3.2 User Interface

Once the system detects a task switch it displays a small popup window in the bottom right corner of the screen (Figure 1). The popup asks the user if he/she has changed tasks and, in the case of a switch, what that new task is. If the user confirms the switch, the system registers it as a successful detection and resets its averages to detect the next task. If the user either cancels or ignores the popup, the system registers it as a false positive. If there is no interaction from the user, the popup automatically disappears after 20 seconds and the program tries to detect the next task switch without resetting its averages. In order to minimize the disturbance to the user, the system was set up so that it would not display more than one popup every 5 minutes. The choice of the 5 minute interval was based on the belief that most users tasks would last for longer that 5 minutes [4]. By using this time out we wished to catch most of the task switches while simultaneously reducing user irritation at frequent interruptions.

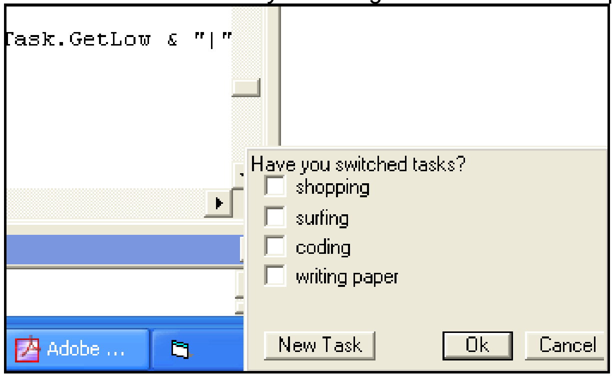

FIGURE 1: A screenshot of the application's task switch confirmation popup window

3.3 Reinforcement Learning

Since users told the system what task they were performing, we were able to use reinforcement learning to tune the threshold limits of the detection algorithm for a given task. If the system correctly identified a task, we applied positive reinforcement and reduced the threshold values for detection; for false positives, we increased the threshold value. The feedback applied was inversely proportional to the number of past reinforcements the system has received. This created damped oscillations in the threshold values before settling around the optimal values.

4. EVALUATION

4.1 Study Design

We evaluated our software by deploying it with six participants for a period of two weeks. The participants consisted of two professors (P1 and P2), three graduate students (G1, G2, and G3) and one IT professional (IP). All the participants were experienced computer users and used Microsoft Windows as their primary operating system. The software was installed on their primary machines for a period of two weeks, during which they were asked to enter the name of their current task whenever the software correctly noted a task switch. The participants were told about the reinforcement learning and were encouraged to train the system by giving it feedback. Once the study was complete, we conducted a semi–structured interview with the participants during which we asked them about their task management styles and the software's effectiveness in picking out a task switch.

4.2 Findings

Of the six participants, all participated for a full two weeks (in some cases voluntarily extending participation to over 3 weeks). By the end of the study, we had detected 1033 task switches (Table 1) from the six users, with 422 (40.85%) being confirmed as accurate. However since the individual subjects used their computers for differing lengths of time during the study, we also calculated the average accuracy across the users (50.12%).

Subject	Switches detected	Switches confirmed	Accuracy (%)
Professor 1 (P1)	57	54	94.74
Professor 2 (P2)	76	43	56.58
Graduate Student 1 (G1)	367	123	33.51
Graduate Student 2 (G2)	217	117	53.92
Graduate Student 3 (G3)	91	37	40.66
IT Professional (IP)	225	48	21.33

TABLE 1: this is another example of a legend

Participants P1 and P2, who perform similar duties, had an accuracy of 95% and 57%, respectively. P1 was very appreciative of the accuracy and said that the software was able to find almost all her task switches. P2, on the other hand, was concerned that the software was unable to detect task switches if he switched between tasks very quickly. He also noted that there were occasions that the software erroneously detected a task switch when he switched windows within a larger task. We believe that the lower accuracy for P2 was a due to a combination of his fast task switching and the 5–minute timeout programmed into the software. Since P2 would often switch tasks within 5 minutes, the popup would appear only after the 5–minute timeout had expired.

The accuracy of detection for G1, G2 and G3 was 34%, 54% and 41%, respectively. The relatively low accuracy for these participants can be partially explained by their use of instant messaging (IM) software. Unlike P1 and P2, these participants were regular IM users and would often chat on IM while concurrently working on their primary task. This complex multi–tasking, along with the bursty nature of IM conversations, led to several task switch notifications being generated due to IM windows. In most of these cases, the participants would often either cancel or ignore the popup because they did not perceive IM as a separate task.

Participant IP was an entry level IT professional who primarily wrote and documented software code. While the window tracking system gave IP the lowest accuracy (21%), he was also its most enthusiastic supporter. In fact, IP was so impressed with the accuracy of the switch detection that he went on to use the log files of the detected task switches to fill out his weekly project time sheets. When questioned about the system, IP said that while it did not miss any task switches, it would sometimes be triggered while switching between windows of subtasks.

5. DISCUSSION

Despite the wide variation in the accuracy, this study has demonstrated the feasibility of using a window manipulation frequency–based approach to detecting task, switches. One of the most important things that we have learned from this study is that the system must adapt to users work habits. The observed variation in detection accuracy can be explained by differences in work patterns. For example the 5 minute time out affected only P2 since he switched tasks much faster that the other users. G1, G2 and G3 were mainly affected by the IM windows and asked to have IM excluded from the window tracking. When questioned about their IM use the users felt that they were not unduly hampered by the notifications – this is contrary to [2], who showed that IM message notifications have a reliably harmful effect on the primary task.

Currently the system cannot distinguish if a user moves to a different task within the same window–the most common example of this is while using web browsers where the user may use the same window for anything from online banking to checking email. We intend to include URL information in future algorithms to detect task switches within a browser. Future work also includes tailoring the window tracking to suit individual users and their work habits. Possibilities include allowing the system to filter out specific windows such as IM and dynamically altering the size of the moving average window and the popup timeout based on user feedback. Dynamic adaptability may also allow this technique to track when users change between subtasks [10] instead of only top level tasks.

6. APPLICATIONS

The potential applications of our window manipulation frequency based task switch detection are extensive. One of the most important applications is in deciding the user's interruptibility at a given time. Research [2, 10] has shown that the best time to interrupt a users is either early on in a task or while switching between sub–phases of a task. Since our system uses only window notification messages it does not have the privacy concerns of [7] while simultaneously avoiding the need for external sensors [3, 6]. The ability to spot a task switch without the need to identify the task itself will also increase user acceptance. Additionally, the computational cost of calculating a simple moving average is much lower than traditional machine learning based techniques.

One unintended application of our system is in supporting time management. Knowledge workers who use a computers as their primary tool and have to keep track of hours spent on specific projects will benefit from a system that can detect a task switch. As shown in the study, subject IP realized this application of the program and co–opted our logs to fill out his time sheets. Other applications of our work include intelligent agents which could use this technique to determine whether a users actions are part of a new task or a part of the current task. This ability to distinguish between tasks will also be very useful in activity support and task archival systems.

7. CONCLUSIONS

Our findings show that it is possible to get highly accurate measurements (over 90% in some cases) of task switches based purely on window manipulation frequency. A two–week user study of software implementing the technique showed that it has great promise and has applications in several fields of research.

REFERENCES

[1] Czerwinski, M., Horvitz, E. and Wilhite, S. (2004) A diary study of task switching and interruptions. In Proceedings of CHI 2004, Vienna, Austria, 24-29 April, pp.175–182, ACM Press.
[2] Czerwinski, M., Cutrell, E. & Horvitz, E. (2000) Instant Messaging: Effects of Relevance and Time. In People and Computers XIV: Proceedings of HCI 2000, Vol. 2, British Computer Society, pp. 71–76.
[3] Fogarty, J., Hudson, S.E., Lai, J. (2004) Examining the robustness of sensor–based statistical models of human interruptibility, In Proceedings of CHI 2004, Vienna, Austria, 24-29 April, pp. 207–214, ACM Press.
[4] Gonzàlez, V.M., and Mark, G. (2004)"Constant, constant multi-tasking craziness": Managing multiple working spheres. In Proceedings of CHI 2004, Vienna, Austria, 24-29 Aprili, pp. 113–120, ACM Press.
[5] Henderson, D. A., and Card, S. K. (1986) Rooms: The use of multiple virtual workspaces to reduce space contention in a window–based graphical user interface. ACM Transactions on Graphics, 5, 3 (July 1986), p. 211–243.
[6] Horvitz, E. and Apacible, J. (2003) Learning and Reasoning about Interruption. In Proceedings of ICMI 2003, Vancouver, BC, Canada, 5-7 November, pp. 20–27, ACM Press.
[7] Horvitz, E., Koch, P., and Apacible, J. (2004) BusyBody: creating and fielding personalized models of the cost of interruption. In Proceedings of CSCW 2004, Chicago, IL, 6-10 November, pp. 507–510, ACM Press.
[8] Hutchings, D.R., Smith, G., Meyers, B., Czerwinski, M., and Robertson, G. (2004) Display space usage and window management operation comparisons between single monitor and multiple monitor users. In Proceedings of AVI 2004, Gallipoli, Italy, 25-28 May, pp. 32–39, ACM Press.
[9] Hutchings, D.R. and Stasko, J. (2004) Revisiting display space management: Understanding current practice to inform next–generation design. In Proceedings of GI 2004, London, Ontario, Canada, 17-19 May, pp. 127–134.
[10] Iqbal, S.T., P.D. Adamczyk, X.S. Zheng and B.P. Bailey. (2005) Towards an Index of Opportunity: Understanding Changes in Mental Workload During Task Execution. Proceedings of CHI 2005, Portland, OR, 2-7 April, pp.311-320, ACM Press.