
A Generalized Interruption Manager

Shawn Sullivan

MIT Media Lab
20 Ames St.
Cambridge, MA 02139 USA
stsully@media.mit.edu

Abstract

Computer users engage their computers in many ways to accomplish work. A variety of tasks, ranging from typing documents to searching the web to writing code, must therefore be able to interact in a consistent, intuitive, non-conflicting manner if they are to permit the user effective time management and cognitive consistency while using the computer. I submit a new interruption management utility which combines aspects of several different research results into a single, multidimensional interruption manager.

Keywords

Interruption, McFarlane classes, task-driven processing, interruption-driven processing, memory load, context aids.

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces: *Interaction styles (e.g., commands, menus, forms, direct manipulation), User interface management systems (UIMS)*

Introduction

The modern pervasiveness of computers has certainly changed the way people work, and provide opportunities for both multitasking and speeding the flow of information. However, computers also present an increased opportunity for users to become distracted by multiple applications competing for the limited resources available on a computer display, not to mention demanding processing time from the limited cognitive ability of the user. I intend to provide a system which first permits the testing of several different interruption styles, and second provides a general system to manage interruption among a variety of applications.

Related Work

Several attempts to mitigate the effects of interruption on the user have been made. With regard to interruption in general, McFarlane provided definitions for four basic types of interruption, each of which has benefits and drawbacks on the user experience [6]. These definitions are referred to herein as McFarlane classes. The four basic categories of interruption are

- immediate interruption, where the interrupting task is allowed to interrupt the user at any time,
- negotiated interruption, where the user is informed of the desire of another task to interrupt him and allowed to choose when to deal with that task,
- mediated interruption, where an intelligent agent decides when to interrupt the user based on an algorithm, and
- scheduled interruption, where interruptions are allowed only at specified intervals.

McFarlane's research suggests each type of interruption has its merits depending on the particular tasks involved in the interruption, indicating that a combination of these different interruption types may provide better results than a system employing just one type of interruption.

Czerwinski et al. have done extensive research on interruption related specifically to instant messaging [2,3]. They have found, among other discoveries, that instant messages, and many other similar types of interruptions, have a greatest effect when occurring during the middle of or near the end of the main task, essentially when the user is executing the task or actively reviewing the state of the task. It is better, they saw, to provide users with interruptions early in the execution of their task because users tend to perform tasks in chunks, known as chunking behavior. Interruptions can therefore be coordinated based on attempting to interrupt only between chunks to improve

task execution speed. Further, they observed in [3] that the interruption of irrelevant tasks is more significant than interruption of tasks related to their current subtask.

Miyata et al. in [5] determined that people work using two basic cognitive processing modes: task-driven and interrupt driven. In task-driven mode, users are intensely focused on a single task. An author deeply involved in actively writing a novel is an example of a task-driven user. In interrupt-driven mode, users are open to switching among several tasks. A person compiling a database using information from several surveys about the traffic through, sales by, and customer opinions of stores in a shopping center is an example of an interrupt-driven user. As the name implies, interrupting a person processing information in an interrupt-driven manner has less significant consequences than interrupting users in a task-driven mode.

Bailey et al. found that tasks with high memory loads are more difficult to resume after being interrupted [1]. Further, they found that interrupting users at task or subtask boundaries, the points at which the user completes one task or subtask and prepares to begin another, are ideal places to interrupt the user.

Finally, Franke et al. researched how system behavior after the interruption has been dealt with can be used to improve the ease with which the user can resume the original task. They found that dividing the process of interruption into three phases, and additionally, a few simple actions can be taken at each phase to reduce the negative impact of interruption. These three phases and the actions suggested for them are

- pre-interruption – a quick rehearsal or review of the state of the current task is useful, especially for tasks with a high memory load
- mid-interruption – supporting the ability of the user to switch back to the original task and maintaining background awareness of the original task are helpful
- post-interruption – providing a recall aid to help the user remember their place in the original task is utile

Purpose

In this paper, we explore a system combining some of the aspects of the research discussed above. Specifically, we examine a system designed to both allow research about the details of individual interruptions and provide an interruption manager for many different interruptions.

Research on the details of individual interruptions may be conducted through using this system to specify how the interruption is to occur. Users can specify the McFarlane class, timing, and other aspects of the interruption easily, facilitating research.

The system may act as a general interruption manager by specifying how several interruptions are to be handled, in the same way individual interruptions are specified for research.

System Design

The system has been developed as described above and implemented for use in a study the interruption of of Open Office Writer (OOW) [8] by AOL Instant

Messenger (AIM) [7], in Microsoft Windows. Interruptions are defined in the format "main_task-interrupt_task," where main_task is the primary task the user is performing and interrupt_task is the task wishing to interrupt the user. So, for example, the interruption of OOW by AIM would be abbreviated the OOW-AIM interruption.

It is of note that the current system assumes each application which may introduce interruptive tasks exists as its own process. This limitation can be removed through additional processing of processor and display usage, or through specific manual coding about individual tasks. However, for the OOW-AIM interruption, this limitation is not relevant because OOW and AIM exist as their own processes.

The basic system allows one to specify several details of the interruption of one process by another. The system collects information about several aspects of the system to determine when an interruption occurs. This information includes

- a list of all processes running on the computer, as well as pertinent details about the process such as memory and processor usage¹
- a list of the status of all open windows on the system and the process with which they are associated, including requests of processes to open windows

¹ This information is collected using PsTools, by Mark Russinovich, a software package capable of observing and manipulating many aspects of a Windows System. See [9] for mor information.

- a list of recent keyboard activity, with the last 2kB (~2000 letters) saved

This information allows a fairly accurate model of the system to be created for identifying interruption. The list of processes and open windows allows the system to recognize possible interruptions as well as identify the current task and which tasks are trying to interrupt, and the saved keyboard activity can be used to identify details of the current and recent tasks and interaction with the interruption. The size of the keyboard buffer can be easily changed by specifying a different size in code, but for the OOW-AIM interruption, 2kB is sufficient.

Individual interruptions are specified by indicating in code what conditions are present at the time of the interruption. These conditions can simply be the presence of a process, or more detailed information based on the details of a process or window. For the OOW-AIM interruption, an interruption is defined as the OOW window being open and active and an attempt by AIM to open a windows. Other designs, such as the simplistic "do both processes exist simultaneously" or a complex design factoring in the process details were attempted, but the design based on the windows was found to be more accurate and flexible with different interruption management techniques.

The system provides several options for how the interruption is to be handled, drawing on the previous research mentioned earlier. The details of the interruption which may be specified and the research related to them are

- The McFarlane class and related details [6]

- Points during the main task that interruption is acceptable [1, 2, 3]
- Whether or not to attempt to determine if the user is in a task-driven or interrupt-driven mode and use that information [5]
- The memory load of the main task [1]
- Whether or not to use context aids before, during, and after the interruption [4]

Figure 1 shows the graphical user interface which permits these configurations, with a sample configuration for the OOW-AIM interruption with an scheduled interruption McFarlane class, interruptions permitted between subtasks and during the beginning of a new subtask, modeling of the type of process used to mediate interruptions, a medium main task cognitive load, and context aids provided after dealing with the interruption. In addition, both tasks use information derived from the status of their windows. Each of these different characteristics affects the interruption in different ways.

The McFarlane Class

The McFarlane class represents the primary method of interruption given to the user. In the absence of other criteria from the other interruption types, this is the method of interruption used. The immediate interruption class simply allows the interrupting task to immediately interrupt the main task any time it needs to.

The negotiated class, upon detecting an interruption attempt, communicates the basic information about the

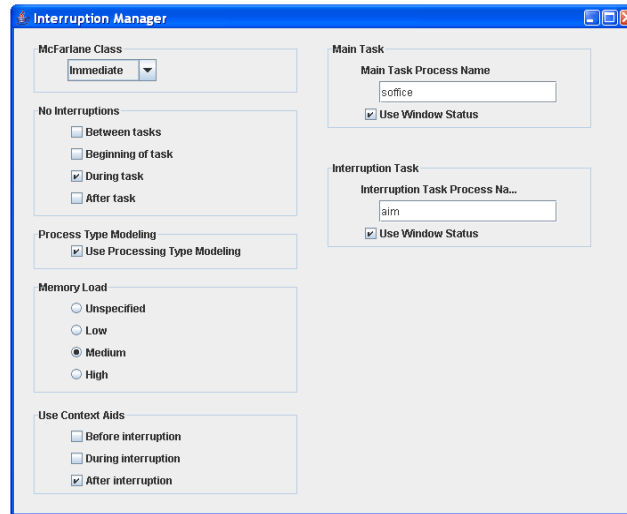


Figure 1: The Interruption Manager Interface

interruption, including the interrupting program's name, and asks the user whether they would like to deal with the interruption immediately, in an inputted number of minutes, when the current task is completed, never, or to ask again in an inputted number of minutes. These rough intervals allow users to switch quickly if not busy, switch after having a quick chance to complete a task or subtask, switch when the current task is completed, identified by the current task no longer being the active window (perhaps a little coarse, but sufficient for the OOW-AIM interruption), never if the interruption is trivial or irrelevant, or to decide later when might be an appropriate time to deal with the interruption.

The mediated class uses a basic task modeler to determine if it is a good time to interrupt the user. The model is rather basic, but effective for many kinds of tasks. It measures the percentage of the previous n

minutes the main task has been the active task (ie, having the window focus) for the user, where n is specified in the interface. If this percentage is below a certain threshold also specified in the interface, it allows the interruption. Note that if the main task has been open less than n minutes, interruptions are more likely, making interruptions at the beginning of a task, a time which has been shown to be easier to interrupt someone. This is a fairly simple model, but when combined with the effects of the other characteristics specified for the interruption, it has more power. Interruptions which are not allowed are queued until an appropriate interruption time.

Finally, the scheduled class permits the interrupting task to interrupt the user once every n minutes, where n is specified in the interface. Note that this is slightly different from the definition used by McFarlane, whose scheduled interruptions were based on allowing interruptions through at specified intervals on the clock, ie, once every fifteen minutes, rather than relative intervals, ie, allow the interruption if there has been no interruption in the past fifteen minutes. Interruptions which are not allowed are queued and given at the next time an interruption occurs, so multiple interruptions can be given during a period, as long as they are simultaneously given to the user.

Interruption Timing

The McFarlane classes, designed by McFarlane as complete interruption management strategies, are treated instead as default strategies; other factors influence interruption. The effect of interruption timing is one such factor. This factor derives from [1, 2, 3] and attempts to allow the user to configure when to be interrupted by the task. They can choose to forbid

interruptions between, before, during, and at the end of the main task by the interrupting task. These choices supersede the McFarlane classes, so for example, if the McFarlane class allows immediate interruption, but interruption is not allowed during the main task, the interruption is queued until it is allowed. When interruption is not forbidden, the McFarlane class is used to control interruption. Providing absolute control over interruption based on timing allows reduces the amount of accuracy required by the McFarlane classes. The mediator does not need to incorporate the the user's status in the task. Some negotiation is essentially handled before the interruption ever occurs, because the user can specify times during a task which he or she knows the interruption will be postponed. The immediate interruption class gains some power and flexibility by not simply allowing all interruptions.

Process Type Modeling

Modeling the type of processing the user is doing is another factor which can be used to add to the McFarlane class. If enabled, before an interruption occurs, the system determines what percentage of the past twenty minutes has been spent with the main task active, or if it has not been open (active or inactive) for twenty minutes, the percentage of time since the the main task was opened. If this percentage is above 60%, it allows the interruption to be managed by the McFarlane class. If not, it queues the interruption until either the percentage falls below 60% or the main task is closed. It should be noted that 60% is an educated guess at the percentage of time a user who is in a task-driven processing mode would use, and can be refined through experiment.

Memory Load

Memory load is used to affect the McFarlane class's behavior, incorporating information with the idea that higher memory load in the main task has a greater cost of interruption. If the memory load is unspecified or low, the McFarlane class is unaffected. If it is medium, the McFarlane class is not allowed to control the interruption until the user has not clicked or typed anything for twenty seconds (however other settings may further preclude the class). If it is high, this time is increased to forty seconds. These delays assume that providing no input means the user is pausing their task, which may not always be an accurate assumption, especially if the user is reading. However, the McFarlane classes also control the interruption, along with the other settings for the interruption, can help mediate this effect. Experimentation to improve these numbers can also help.

Context Aids

The use of context aids affects more the manner in which an interruption is executed, not the timing of the interruption. Context aids help orient the user throughout an interruption, and are more sensitive to the behavior of the specific tasks involved because they relate to the way in which applications display information, rather than general ideas about when interruption is appropriate. Thus, selecting the use of context aids before, during, or after the interruption only indicates to the system to attempt to find in its code an action to take at the appropriate point in the interruption.

As an example, consider the context aids used in the OOW-AIM interruption. The context aid before the interruption is to highlight the previous two sentences

to allow the user to quickly see what they were just working on. During the interruption, the interrupt manager ensures that the OOW remains full screen , with only the AIM window, which is much smaller than full screen, over it, allowing the user to continue to see their work in OOW. After the interruption, the interruption manager positions the view in OOW such that at least two-thirds of the view is text the user has already written, and highlights the user's current paragraph. These behaviors are hard-coded, and currently all such behaviors must be hard coded using a simple API.

Window Status

In addition, the main task and interrupting task may be individually configured to use window information, which simply means that the applications use the information about their associated windows in addition to information about their existence as processes. In general, tasks which involve windows should use window information, or certain interruption management facilities will be limited.

Experimental Design

The above system has not been formally tested. However, an appropriate two-stage experiment has been designed to test its effectiveness.

The first experiment is designed to determine the effectiveness of the system in mediating the OOW-AIM interruption versus pure McFarlane class management. 44 users, 22 males and 22 females, with OOW and AIM familiarity will be collected and told to write an essay on a thought-involving essay, such as why they are what religion they are or why they support or dislike the president, in 45 minutes using OOW. During this

time, they will receive five interruptions through AIM, requiring various amounts of concentration to resolve. Four subjects will have no interruption management as a control. Four subjects each will be given pure McFarlane class management, a total of sixteen subjects, for the four classes immediate, mediated, negotiated, and scheduled. Finally, six subjects each will be given a version of this system for each McFarlane class, with additional interruption management provided from the other various settings. The results will provide data which will show the effectiveness of various interruption management aspects added by this system on the OOW-AIM interruption, and probably allow some degree of generalization on the effectiveness of this system.

The second experiment is designed to be a qualitative study of the overall effectiveness of the system as an interruption manager. After a series of preliminary studies to specify some effective management strategies for individual interruptions for many common tasks, a system will be created which incorporates all of them into a single interruption management strategy. This system will be distributed to at least five people for evaluation over a week of actual usage. The results of this experiment, and perhaps a few related followups, should allow for the creation of a general interruption manager which can be quantitatively measured.

References

1. Bailey, B.P., J.A. Konstan, and J.V. Carlis. Measuring the Effects of Interruptions on Task Performance in the User Interface. *IEEE Conference on Systems, Man, and Cybernetics*, 2000.
2. Czerwinski, M., Cutrell, E. & Horvitz, E. (2000). Instant Messaging and Interruption: Influence of Task Type on

Performance, In *Paris, C., Ozkan, N., Howard, S. and Lu, S. (Ed's.), OZCHI 2000 Conference Proceedings*, Sydney, Australia, Dec. 4-8, pp. 356-361.

3. Czerwinski, M., Cutrell, E. and Horvitz, E. (2000). Instant Messaging: Effects of Relevance and Timing. In S. Turner, P. Turner (eds), *People and Computers XIV: Proceedings of HCI 2000*, Vol. 2, British Computer Society, p. 71-76.

4. Franke, J. L., Daniels, J. J., & McFarlane, D. C. (2002). *Recovering Context After Interruption*. Paper presented at the 24th Annual Meeting of the Cognitive Science Society, Fairfax, VA.

5. Miyata, Y. & Norman, D. (1986), Psychological Issues in Support of Multiple Activities, in D. A. Norman & S. W.

Draper (eds.), *User Centered Systems Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, pp.265-84.

6. McFarlane, D. (1999). Coordinating the Interruption of People in Human-Computer Interaction. *Proceedings of the IFIP TC.13 Conference on Human Computer Interaction* (Interact 1999).

7. AOL Instant Messenger, <http://www.aim.com>.

8. Open Office, <http://openoffice.org>

9. PsTools by Mark Russinovich,
<http://www.sysinternals.com/Utilities/PsTools.html>