A Formal Approach Supporting the Comparative Predictive Assessment of the Interruption-Tolerance of Interactive Systems

Philippe Palanque, Marco Winckler, Jean-François Ladry
[1]IHCS-IRIT, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse, France
{palanque, ladry, winckler}@irit.fr

Maurice H. ter Beek, Giorgio Faconti, Mieke Massink
ISTI –CNR, Area della Ricerca, via G. Moruzzi 1, 56124 Pisa, Italy
{massink, faconti, terbeek}@isti.cnr.it

ABSTRACT
This paper presents an approach for investigating in a predictive way potential disruptive effects of interruptions on task performance in a multitasking environment. The approach combines previous work in the field of interruption analysis, formal description techniques for interactive systems and stochastic processes to support performance analysis of user activities constrained by the occurrence of interruptions. The approach uses formal description techniques to provide a precise description of user tasks, and both system and interruptions behavior. The detailed mechanism by which systems and interruptions behave is first described using a Petri nets-based formal description technique called Interactive Cooperative Objects (ICO). The use of a formal modeling technique for the description of these three components makes it possible to compare and analyze different interaction techniques. In particular, it allows us to determine which of the system states are most affected by the occurrence of interruptions. Once composed together, models describing the system, user tasks and interruptions behavior are transformed into PEPA models (i.e. Performance Evaluation Process Algebra) that are amenable to performance analysis using the PRISM model checker. The approach is exemplified by a simple example that models two interaction techniques for manipulating icons in a desktop environment.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: Evaluation/methodology; Interaction styles (e.g., commands, menus, forms, direct manipulation).

General Terms
Human Factors, Performance

Keywords
Model-Based approaches, formal description techniques, interruptions, performance evaluation.

1. INTRODUCTION
Many working environments involve the occurrence of multitasking and require users to temporarily suspend a task to complete an unexpected intervening activity [28]. In multitasking systems, interruptions are considered as simple breaks in the current task execution that causes a (unexpected or intended) deviation in the control flow. However, research in the field of Human Computer Interaction (HCI) has shown that efficient task shifting does not imply that users will be able to suspend and resume tasks efficiently [19]. Some empirical studies try to elucidate the effect of task switching and the disruptive effects they have on user activity during the interaction on computer tasks [7][11][16][29]. It has been demonstrated that interruptions can also lead to incidents related to human error. For example, pilots experiencing interruptions during preflight checklists have been blamed for multiple aviation crashes [13][26]. More recent studies have shown that interruptions may be an important factor in driving [24] and emergency room care [8].

Current literature about interruptions in human computer interaction addresses this problem from one the following different perspectives: a) psychology of human interruption [29]; b) technologies for improving the quality of interruption generation [23]; c) HCI methods for brokering interruptions [17]; d) the effects of interruptions in work settings [28]; and e) case studies describing the results of introducing technologies into the workplace in an attempt to improve coordination performance [25].

The work presented in this paper introduces a new perspective for the research in this field based on model-based analytical techniques for investigating potential disruptive effects of interruptions on task performance in a multitasking environment. A system A is called more interruption-tolerant than a system B if, for the same rate and type of occurrence of interruptions, the number of goals reached by a given user is higher than with the system B. In the current paper interruptions are considered as an independent system (called interruption source) and we do not consider other types of interruptions as multi-threaded dialogues. Our approach combines previous work on the field of interruption analysis, formal modeling of interactive systems and stochastic processes to support performance analysis of user tasks constrained by the occurrence of interruption in the working environment. The approach uses formal description techniques to provide a precise description of user tasks, system and interruptions behavior.

2. STATE-OF-THE-ART
Several empirical studies try to elucidate the effect of task switching and the disruptive effects they have on user activity during the interaction on computer tasks (for a comprehensive review of the literature, see [29] & [35]). Notwithstanding we report some of the most interesting findings that are relevant for our study.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EICS'09, July15–17, 2009, Pittsburgh, Pennsylvania, USA.
Copyright 2009 ACM 978-1-60558-600-7/09/07...$5.00.

2.1 The Inner Nature of Interruptions

Task interruption in the workplace is a widespread phenomenon. However, to better understand the effects of interruption on human activity it is important to have a clear understanding of the steps leading to task shifting. According to Trafton and Monk [35], interruptions occur when a person is working on a primary task (usually long-lasting) and an alert for a secondary task occurs. An important aspect of alerts is that they create an interruption lag as the user has to turn their attention to the interruption. The person then starts the secondary task. After completing it, the person must resume the primary task. During the resumption lag, the person must figure out what they were doing during the primary task and what to do next. Finally, the person resumes the primary task. From this task analysis and real-world example, it is clear that different aspects of the cognitive system are relevant to the study of interruptions and resumptions.

Sources of alerts could be either internal (i.e. user thoughts) or external (e.g. a person coming into the room to ask the person a question, a fire alarm, instant messaging). Internal interruptions are very difficult to detect; is some cases they might lead to task abandons that should be considered normal deviations on the user scenario, for example when the user gives up trying to reach the initial goal. The external sources are numerous and vary, ranging from social events from the environment to system alarms. However, as far as interruptions are raised in a computer environment, some strategies can be employed to decide when to interrupt someone during multitasked computing [23].

The understanding of user behavior during each phase of the interruption lifecycle raises questions of theoretical and practical importance. Iqbal and Horvitz [19] argue that according to the phase in the interruption's life cycle, different strategies can be employed to assist computer users with multitasking shifting.

2.2 Interruptions Disruptiveness

Research has shown that different types of interruptions can affect their disruptiveness [3]. Quite often, interruptions are associated with negative effects: resuming a task after an interruption is difficult and may take a long time, interrupted tasks are perceived as harder than uninterrupted ones, interruptions cause more cognitive workload and they are quite often annoying and frustrating because they disrupt people from completing their work. Indeed, frequent interruptions can reduce user performance, however not all interruptions bring negative impact: awareness systems such as alarms and alert systems effectively shift our attention to matters that need immediate care [23] and interruptions may actually increase performance [34].

Gillie and Broadbent [16] showed that being able to rehearse one's position in the main task does not protect one from the disruptive effects of an interruption. They found that interruptions with similar content could be quite disruptive even if they are extremely short.

Other researchers have also studied the timing of interruptions, and how an alert can allow a person to anticipate an interruption [13, 25, 18]. Alerts essentially create an interruption lag and results of these studies have shown that an interruption lag can reduce the disruptive effects of interruptions, primarily by reducing reorientation time to the primary task after the interruption task is completed and thereby reducing overall performance time of the primary task. Interruption lags in these studies allowed participants to either finish what they were working on before attending to the interruption, or encode retrieval cues to allow for better task resumption following the interruption.

2.3 Surviving Interruptions

Although the research on interruptions is still relatively new, and much work still needs to be done at both theoretical and applied levels, there is some evidence on how to make interactive systems more resilient and how to reduce the disruptive effects on user tasks. If people are simply learning the task; training people on the task itself would reduce the disruptiveness of the interruptions [35]. However, if people are learning how to resume, then training on interruptions and resumptions should be built into current training practices. Trafton & Monk [35] found that interruptions became less disruptive over time with experience and practice on the resumption process itself; experience on the primary task alone (without interruptions) did not reduce the disruptiveness of interruptions. This study strongly suggests that for training people in complex domains, training scenarios should include occasional naturalistic interruptions in order to reduce the disruptiveness of interruptions during actual performance.

Research on task-switching processes has shown that people struggle to remember to switch between operations. In order to improve user performance it has been suggested the use of cues signaling when the secondary task requires attention [17]. This suggests that providing an external mnemonic may greatly benefit performance for people who deal with safety critical interruptions and prospective memory tasks. Using the Long Term Working Memory (LTWM) perspective, Oulasvirta and Saariluoma [29] also made several applied suggestions. Based on the results of their experiments, they suggested that system designers should keep the sequence of user actions as short as possible. The amount of time does not seem to be theoretically determined, but 20 seconds seems to be a heuristic used by some designers. They also suggested preventing interruptions on tasks that require large amounts of encoding time (e.g., certain checklists that airline pilots go through).

Some tools like GroupBar developed at Microsoft Research [11] have been specifically designed to allow people to save and retrieve application and window management setups, which can be extremely useful when switching tasks. Iqbal and Bailey [20] also have built several tools based on empirical work. They found that the best moment to interrupt people is between breakpoints between tasks. They have created a tool that is able to automatically detect times of high and low workload and it is able to interrupt people at times of low workload.

2.4 Modeling Task Interruptions

There have been several attempts to formalize cognitive models describing the impact of interruptions on human behavior [35]. However, only a few have addressed formal description techniques to describe the occurrence of interruptions in system specifications [21].

In multitasking environments, interruptions should be seen as just as a break in the current task execution that causes a (unexpected or intended) deviation on the control flow. This problem is quite well known in the domain of Operational Systems where the occurrence of interruptions during execution of programs running in parallel presents many similarities with multitasking in human activity. Previous work in the Operational System domain has demonstrated that systematic description of all deviations on the

system control flow is almost impossible as it would lead to an exponential and unpredictable number of states.

There are some situations where the interruption of a task should be considered as part of the user goals as, for example, to cancel document printing. Various notations are currently available for modeling tasks [12]. A notation belonging to this category is Concur Task Tree (CTT) [33]. Indeed, CTT explicitly provides a means to represent task interruption by the means of the *interruption* operator (i.e. |>) as presented by Figure 1. However, operations describing interruptions in tasks models should only be used when the tasks interruption belong to a user goal (here it is a possibility offered to the user to cancel the printing). Similarly to CTT, West and Nagy [36] have added theoretical structures to the notation GOMS in order to overcome its limitations for analyzing interruptions when task switching are common.

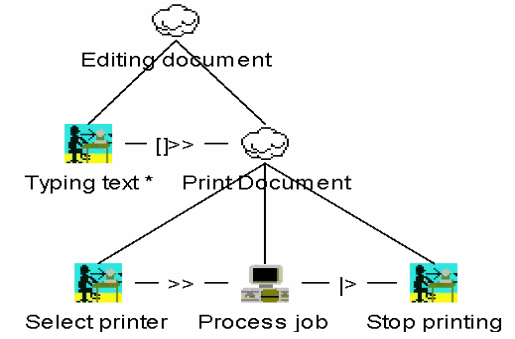

Figure 1. Interruption operator "|>" in CTT.

Jambon's work [21] extends UAN (User Action Notation) with additional relations that takes explicit into consideration different manifestations of interruptions such as for example interruptions with interleaving, with parallelism, without task resumption, and with resumption at a different point. In addition, there is more recent work on interruptions that is relevant to our work [10] and [2] where the focus is on system malfunctions due to human actions. This work instantiates a generic cognitive architecture to obtain specific user models with an inspiration very similar to syndetic modeling; the major difference being that they take an even more abstract view of cognitive processes, since they focus on goals and task knowledge of users to detect potential interaction problems due to conflicts with this knowledge.

While all this related work is highly relevant to our work, the current paper mainly addresses the issues related to modeling behavioral aspects of interruption in order to be able assess in a predictive way the impact of interruptions on task performance i.e. without (or prior to) carrying out user studies.

3. THE APPROACH

This section proposes a design process for the design and assessment of interruption-tolerant systems. Figure 2 presents the iterative process of the construction of models using formal description techniques. In the beginning of the process a preliminary model is constructed that is then analyzed in order to assess what kind of properties it fulfils. According to the result of the analysis process it can be decided that the model has to be modified. In [31] the interested reader can find a detailed explanation on how to perform such verification in the field of interactive systems, in order to deal with that complexity.

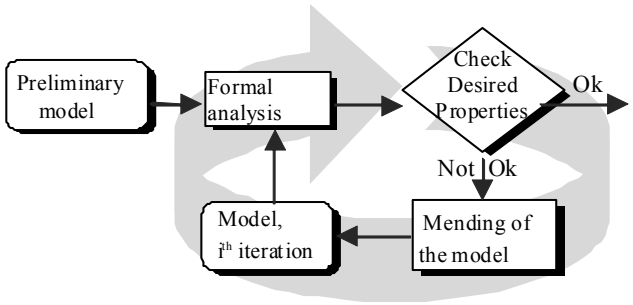

Figure 2. Iterative process for building a model

As the ultimate goal of a safety critical system is to allow operators to perform their goals in an efficient and error free manner, we have previously presented a process integrating tasks modeling and system modeling. The process made it possible to assess the compliance of the tasks and system models [32]. That paper presented, on a case study, how iterations are performed, how models compatibility can be checked and what kind of properties can be proven. The current paper extends that previous work to exploit performance evaluation technique to compare the interruption-tolerance of systems.

The basic assumption, in this design process, is that the interruptions are considered as another independent system (called Interruptions Source) located in the environment of the interactive system under consideration. The Interruption Source is thus modeled as a system per se. Here again, the modeling activity starts with a preliminary model that is subject to modifications according to the analysis results or to further understanding of the behavior of the Interruptions Sources. Indeed, while we imply that there is only one interruption model there can be several whose behavior are integrated in a single model. It is important to note that (most of the time) the model of the Interruptions Sources only triggers elements of the system and tasks models. However, it can also be the case that the model of Interruptions Sources can be accessed from the system model. This can only be the case if one of the Interruptions Sources provides an (Application Programming Interface) API for interfering on the predefined behavior of the Interruption Source.

3.1 Modeling Activities

Figure 3 presents the extended process exhibiting system, tasks and interruption models.

A critical point in the process is the consistency assurance activity (centre of the diagram). Indeed, it is important to ensure that the three models are compliant as they represent 3 different views of the same world i.e. the operator (tasks model) interacting with a system (system model) while external sources (Interruption Sources model) can interfere with that activity.

In the safety critical domain, Interruptions Source can be both computing resources (such as an alarm (as, for instance, a Ground Collision Avoidance System (GCAS) in a cockpit) and socio-technical resources (as an Air Traffic Controller sending a clearance to the pilot of an aircraft). For sake of brevity, and as it does not have an impact on the approach we propose, we consider that interruptions are only part of an external system model and will thus be modeled independently.

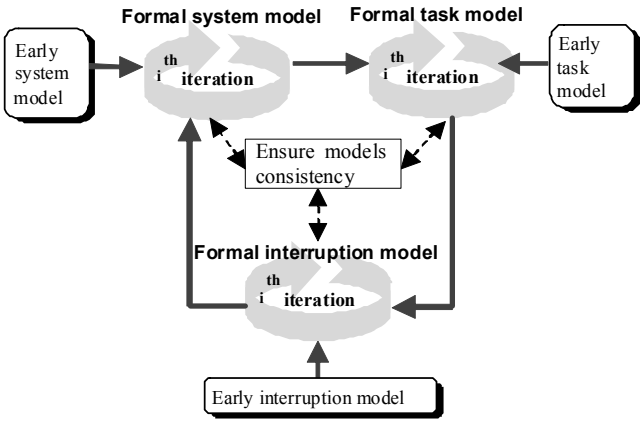

Figure 3. Iterative process involving system, tasks and interruptions models

As the goal of the approach is to assess the interruption-tolerance, Figure 4 presents the process for performing such assessment on two different systems and in a relative way. The two boxes on the right-hand side of the Figure correspond to them (system A on top and System B on the bottom). Both systems are aimed at supporting the same user goals and at receiving the same perturbations from the Interruptions Sources and, consequently, the interruptions model is the same in both cases.

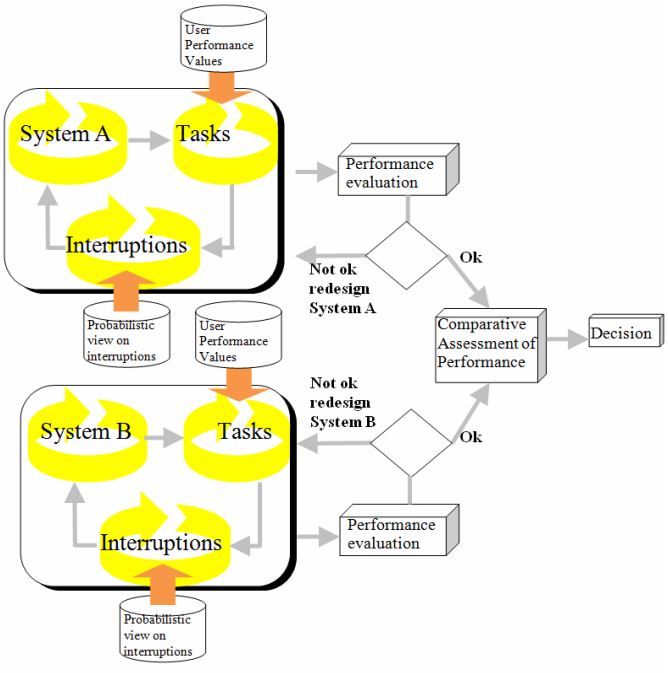

Figure 4. Process for assessing the impact of interruption on user performance for two different systems

3.2 Values Injections

In order to be able to compute performance evaluation of the systems, performance values are inserted in the models.

User performance values and cognitive theory are used to enrich and extend the task model to obtain a comprehensive model of user behavior according to the type of the tasks (motor, cognitive or perceptual). Such values correspond to information available in the field of HCI such as Fitts' law data for motor values [15], Human processor for cognitive and perceptual performance values [6] and ICS (Interacting Cognitive Subsystems) [4]. For instance Human processor model states that cognitive processor cycle time has a mean of 70 ms (max-min being 25-170 ms). Other, more recent data can also be used as the field is evolving quickly and more research is regularly performed especially providing data for prediction related to interaction techniques as [1] for constraint movement and [9] for zoomable interfaces.

The validity of such data is critical when performance evaluation of the triplet (system, tasks (extended with user values), interruption) is concerned. This phase is represented by the boxes labeled "performance evaluation" in Figure 4 and 5. The results of the phase will be used as a re-design driver in order to decide (in the case of performance lower than required/expected) to modify the elements: the system (by for instance by changing the interaction technique) the tasks (for instance by modifying the training of the operators and, by consequence the tasks they have to perform), the interruption (if it is possible to influence that "external" element).

However, it is important to note that the exact value of data is not important per se, as far as the comparative assessment is concerned, and the same values are chosen to model activities that the interaction techniques have in common. Indeed, such assessment is relative i.e. does not provide an absolute measure of interruption-tolerance but a comparison of two systems (see "comparative assessment of performance" box in Figure 4). Therefore, as the same (sub-)tasks are modeled by the same values, the actual values have no influence on the result of the assessment. This argument however does not hold if the interaction techniques are not similar (as in the example chosen presented in section 4).

3.3 Instantiation of the approach

Figure 5 presents the instantiation of the process of Figure 4 including the notations and the values presented above. The light grey circles contain the underlying theories from which data are produced and the notations used for building the various models. For instance System and interruption models are described using the ICO notation.

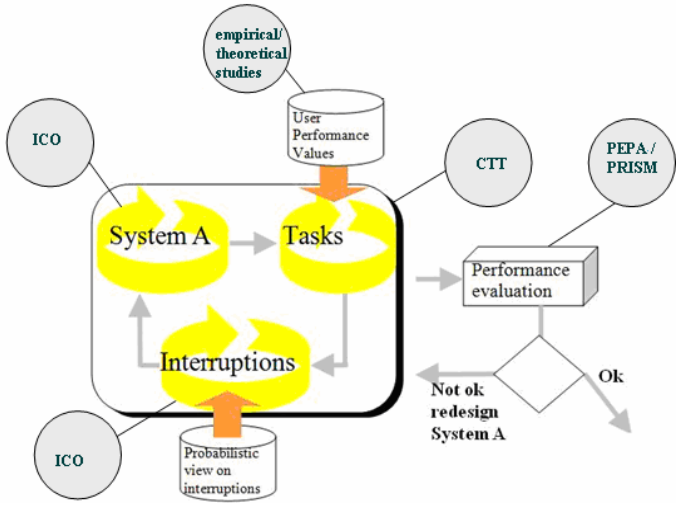

Figure 5. Instantiation of the Process of Figure 4

Performance analysis is performed using the PRISM model checker. As it requires, as input, stochastic automata, a transformation needs to be defined from both ICO and enriched CTT

specification into the PEPA process algebra. At this stage the process remains theoretical making it applicable to a large set of interactive systems from various application domains.

The next section sketches how this approach can be instantiated on a case study, presenting the models, the transformations and the results of the performance evaluation.

4. CASE STUDY

In order to exemplify the approach described above we introduce a simple case study. The objective of the case study is to present the various phases of the approach on a simple but realistic application. Figure 6 presents the user interface of the case study. In the application a set of icons are presented in a window on a grid. The icons can be moved to different locations and deleted.

The user's tasks are thus limited to the one presented in Figure 7. The user's goal (upper task in the tasks model) is to remove all the icons on the user interface. This can be achieved by doing, in any order, the selection of an icon and the triggering of a deletion command. In order to reach the goal, users have to perform iteratively the selection and deletion of icons (represented in the model by the * symbol next to the upper abstract task "Clear Icons".

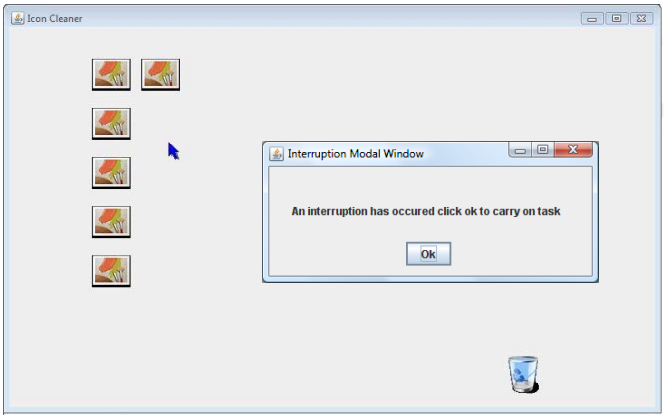

Figure 6. User Interface of the case study

To support this goal two different systems have been constructed. System A (called Drag & Drop) features an interaction technique of type Drag and Drop. Icons on the user interface can be selected by moving the mouse cursor over them and pressing the left button. Once selected, icons can be dragged in the window at any new position. If the mouse button is released when the icon is positioned over the icon of the trash, then the icon is deleted. System B (called Speak & Click) features a multimodal interaction technique involving speech recognition (for the deletion command) and gesture (for icon selection). Systems and tasks models related to these two systems are presented in the next sections.

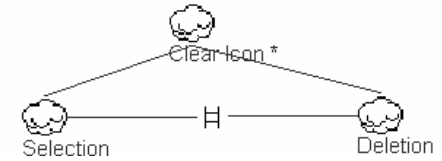

Figure 7. Abstract tasks model expressed in CTT

4.1 Modeling System A (Drag & Drop)

The behavior of System A has been fully modeled using the ICO notation and is presented in Figure 10. Indeed, the model contains all the preconditions about the current state of interaction as well as the set and sequences of events that are available to the user.

According to Figure 3 system and task models need to be kept consistent. As shown in the Figure the system has an impact on how the tasks can be performed by the user. Thus going from an abstract description of the tasks to a detailed one (fitting with a given system), the tasks models are different. According to the description of System, the abstract tasks model presented in Figure 7 has to be refined as presented in Figure 8. Indeed, Selection and Deletion tasks can be now refined more precisely. Selection is performed first by deciding the icon to be deleted then by moving the mouse cursor on the icon and by pressing the mouse button.

Deletion is performed by moving the selected icon over the trash icon, verifying that trash icon is highlighted and releasing mouse button.

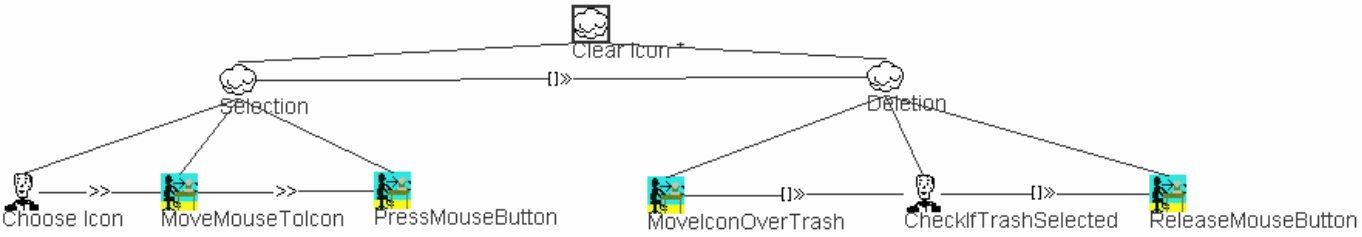

Figure 8. Task model refined to be conformant with Drag & Drop behavior (system model A)

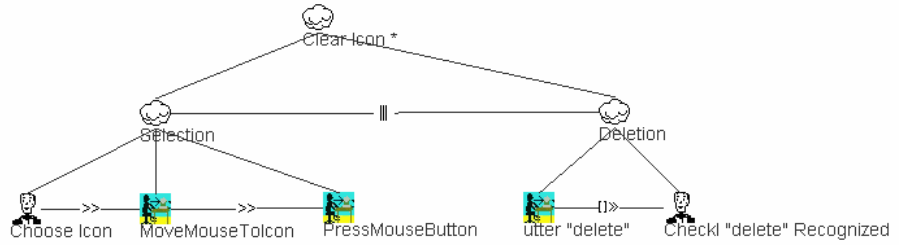

Figure 9. Task model refined to be conformant with Speak & Click behavior (system model B)

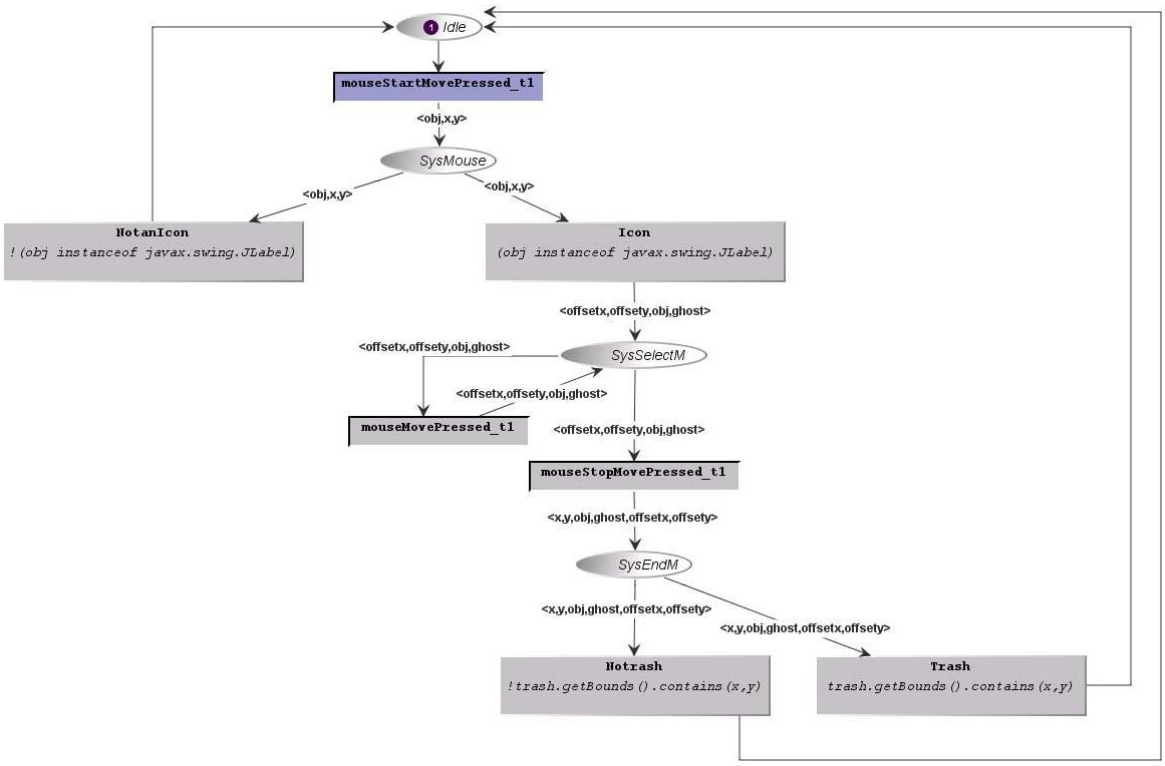

Figure 10. ICO model of system A (Drag & Drop interaction technique)

It is interesting to note that the temporal operator *order independence* between tasks Deletion and Selection is more constrained in the refinement as it has been replaced by a *sequence* operator. Adding constraints is allowed in the refinement process, as what is allowed in the refined model remains compliant with the abstract description. Relaxing constraints is not allowed as it would make it possible for the users to trigger sequences of action (in the refined model) not accepted by the abstract model.

4.2 Modeling System B (Speak & Click)

As for System A, System B has been fully described using the ICO formalism and is presented on Figure 11.

This model allows users to either start by a speech command "delete" and then selecting an icon or selecting first an icon to be deleted and then uttering the word "delete". That model is a refinement of the ICO model describing the application. It includes the interruption sources behavior and excludes manipulation errors or recognition errors. In any case, interaction must start either by the user moving the mouse (this is represented by the transition **move** on the upper part of the model) or by uttering a word (**start-speak**).

As has been done for System A, the abstract tasks model presented in Figure 7 has to be refined in order to be compliant with System B. Figure 9 presents the refined task model. It is interesting to note that this task model is less constrained as it accepts Deletion and Selection in any order (modeled by using the *concurrent* operator at the first level).

4.3 Modeling Interruptions

According to the approach presented in Figure 3 the last model to be built is the model describing the behavior of the Interruption Sources. While the state of the art section has explained the anatomy of interruptions and the various research results (especially in psychology about the impact of interruptions on users' activity) we present in this section the simplest model of interruptions possible. Indeed, this paper is devoted to the description of a process integrating tasks, systems and interruptions models and how the resulting models can be used to compute performance evaluation on that triplet. We are currently integrating this process in the area of interactive cockpits, extending previous work in the re-configuration of cockpit applications when part of the hardware side of the user interface has failed [28]. Indeed, such failures and re-configurations are intrinsically perceived by the operators as interruptions and assessing pilot performance in such a context is critical.

The interruption taken into account here behaves as follows. Each time an interruption occurs a modal window occurs featuring an Ok button (see window in the center of Figure 6) is displayed and the user has to click on it to be able to carry on the initial task. Figure 12 presents such behavior using the ICO notation. In the initial state the interruption model is idle (there is a token in the place *Interrupt*) waiting for an interruption to occur.

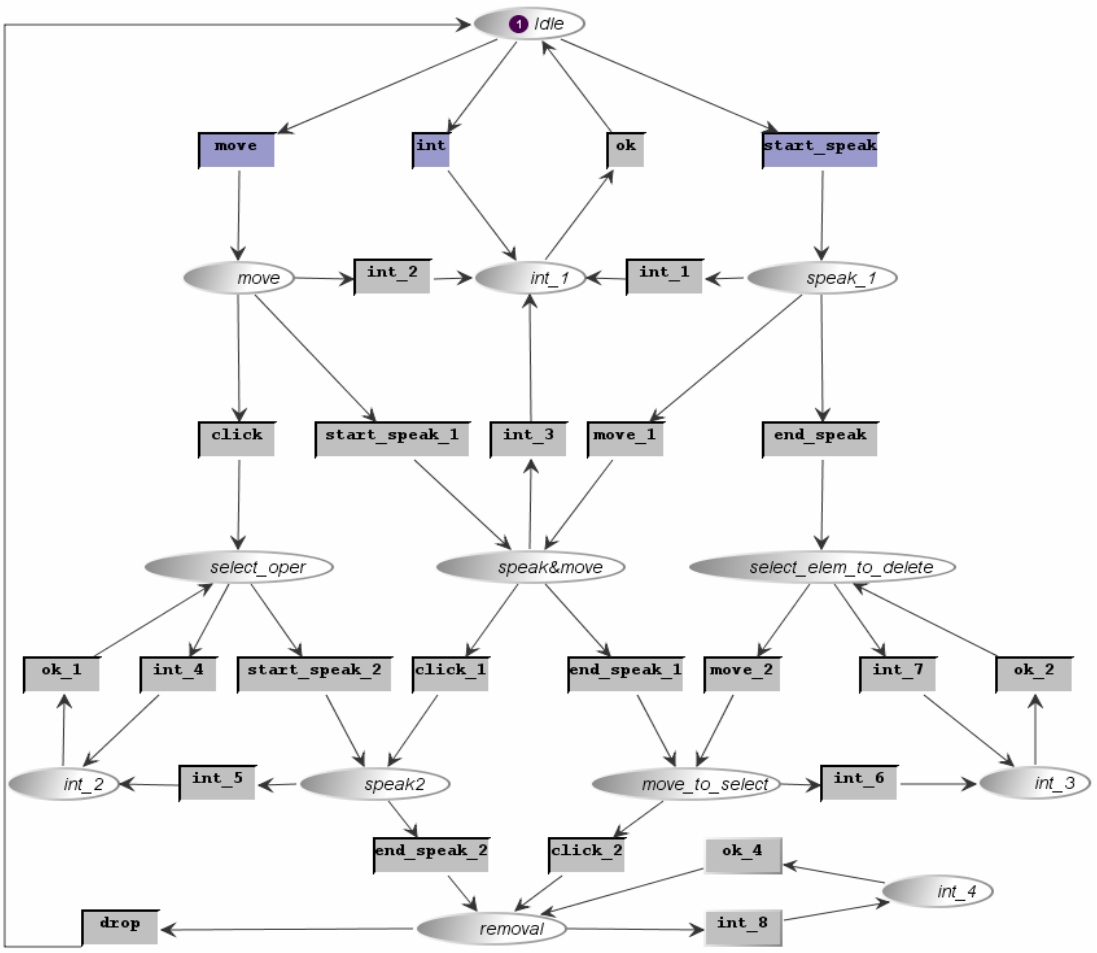

Figure 11. ICO model of system B (Speak & Click interaction technique) composed with the behavior of the interruption source

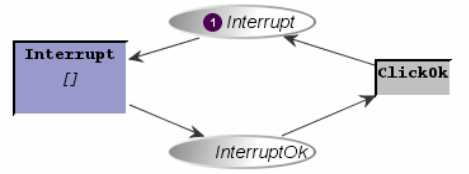

Figure 12. ICO model of Interruption Source

This is modeled by transition **Interrupt** which is triggered according to a temporal evolution (represented by the symbol []). When the interruption has occurred, the token in place *Interrupt* moves to place *InterruptOk* and the transition **ClickOk** becomes available. This transition is connected to the Ok button i.e. when the event is triggered by the user, the transition is fired and the model comes back to its initial state.

4.4 Transformation to Stochastic Automata

The next step in the process is twofold: first to transform each of the models from ICO and enriched CTT to a PEPA specification, then to specify their composition by means of the PEPA *cooperation* operator thus enabling performance evaluation with the PRISM model checker.

To respect page constraints we will only present the result of that part of the transformation concerned with the interrupt model. Currently this transformation has been done manually but tools could be developed to support that activity. This is indeed requested for future work as the manual nature of the transformation might introduce deviations between the ICO system models corresponding to the running application. Figure 13 presents the stochastic automaton produced from the ICO model presented in Figure 12. In such simple models, transformation is trivial but it can be more complex in the case of ICOs with a large state space (for instance having several tokens in the same place) or featuring several concurrent constructs to be unfolded in an automaton. Automaton states are indentified by using the marking graph of the Petri net. This marking graph corresponds to the set of possible markings in the net i.e. the entire possible distribution of token in the places of the net.

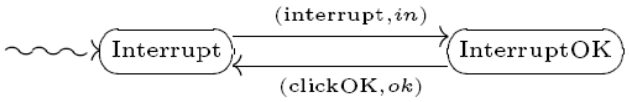

Figure 13. Automaton produced from model in Figure 12

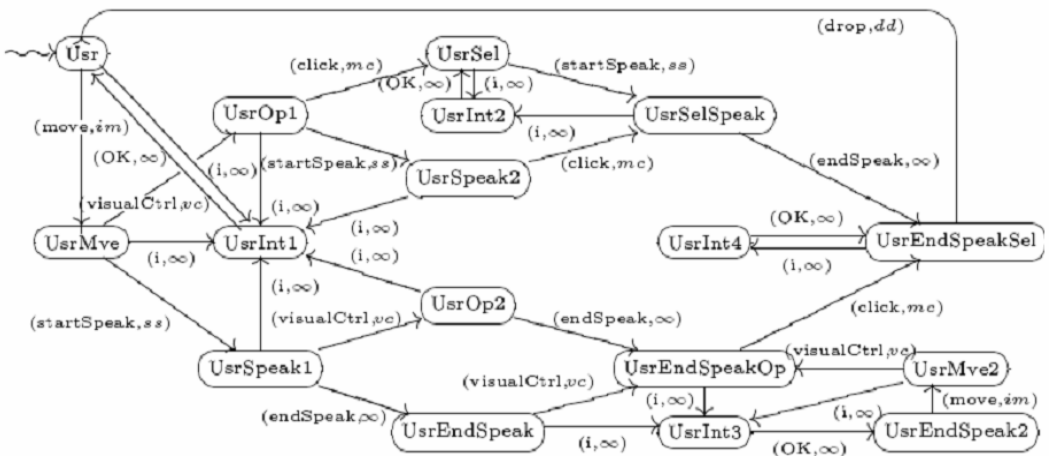

Figure 14. Automaton of user behavior receptive of interruptions for system B. For readability clickOk=OK and i=interrupt

According to the approach used for assessing the performance each ICO model has to be transformed into a dedicated automaton. These automata are then composed using the PEPA *cooperation* operator to define the action types on which the components must synchronize (cooperate). This part of the specification is not presented here due to space constraints. However, the interested reader can find detailed information about PEPA and the case study in [5].

The production of the automaton corresponding to the ICO model in Figure 11 is facilitated by the structure of that particular Petri net. Indeed, this Petri net is a state machine as each transition in the model has only one incoming arc and one outgoing arc [31]. For this reason the model is not represented here.

4.5 Enriching Tasks Models: User Behavior Automaton

The automata describing the performance of the tasks, are explicit models of the user derived from a cognitive theory (ICS in our case), specialized to perform the intended tasks, and parameterized with respect to the mean rate of occurrence of actions. For example, the model presented in Figure 14 describes the activity of a user performing a speak & drop type of interaction as described by the corresponding task model of Figure 9, augmented with the capability of dealing with interruptions. This user is able to perform a speak & drop interaction by using speech and gesture in parallel if and only if she's able to start speaking before focusing on the selection of the icon. On the contrary, the interaction is forced to be a sequential one due to insufficiency of cognitive resources if the user first focuses on selection of the icon. This is represented in the automaton with the choice between the actions (*startSpeak, ss*) and (*visualControl, vc*). Error behavior is not modeled in order to separate the effect of errors from that of interrupts in the analysis. In fact, we can see that all the possible paths in the model lead to the final action (*drop, dd*) corresponding to the actual removal of the selected icon from the display (i.e. reaching the goal).

4.6 Injection of values

The last activity to be carried out prior to performance evaluation is the assignment of rate values to transitions. For instance, in Figure 13, the variable *in* of the transition labeled (*interrupt, in*) models the mean rate of the exponential distribution at which the triggering of the modal window occurs. The variable *ok* appearing on the transition label (*click, ok*) corresponds to the mean time of user reaction (time to click on the Ok button of the modal window).

Similarly, the principle for value injection in the models is only carried out using mean rates for execution or cognitive actions. For example, these values can be obtained from empirical observation (when experiments are available), theoretical models (such as KLM, ICS, Fitts' Law, etc) or a range of values used to support the hypothesis one wants to investigate (e.g. what is the performance for varying interrupt rates).

Table 1. Values used in our case study.

Symbol	Mean time (ms)	Meaning
1/ss	630	start speaking & completing the "delete" utterance
1/es	1000	end speaking (plus recognition and feedback)
1/im	910	user planning + ballistic movement
1/vc	290	visual control including both approach and adjustment phases of movement
1/mc	120	mouse click
1/dd	120	icon removed
1/ok	1300	Interrupt served
∞		process is cooperation passive on the corresponding action

All these values shown in Table 1 are represented on the labels of the transitions of the automaton. As the current paper focuses on *the* procedure and not on the performance evaluation per se, we only present the values used in the performance analysis omitting a detailed justification which can anyway be found in [5]. The first column of the table corresponds to the symbols appearing in the transition label of the automaton in Figures 13 and 14.

4.7 Comparative Assessment of Performance

As an indicator of the interruption-tolerance of an interaction technique to external interrupts, we study the number of effective drops (corresponding to the deletion of the icon by the system) a user manages to perform during a fixed period of time under a varying number of interrupts occurring randomly during that period. The expected number of drops and interrupts occurring during an interval of 5 minutes (i.e. 300s) are cumulative reward

measures that can be formalized in reward stochastic temporal logic supported by the PRISM model checker.

Figure 15 summarizes the expected number of drops a user manages to perform in the presence of a number of interrupts over the time span, for the Drag and Drop and for two different implementations of the Speak and Drop interaction techniques. The two implementations of Speak and Drop differ in the speech recognition engine: in one case the recognizer operates at 2.5 x real time, as it happens in the case of general speech recognition, in the other case the recognizer operates in real time, as it is the case for specialized systems. With an interrupt rate close to 0, the user performs on average (given the values chosen for the model's parameters) 124 drops when using Drag and Drop, only 109 when using Speak and Drop, and 158 when using the real time recognition engine. As expected, the user's performance decreases when the interrupt rate increases. In the presence of about 130 interrupts in 300s, the user manages to perform only 23, 33 and 55 drops respectively.

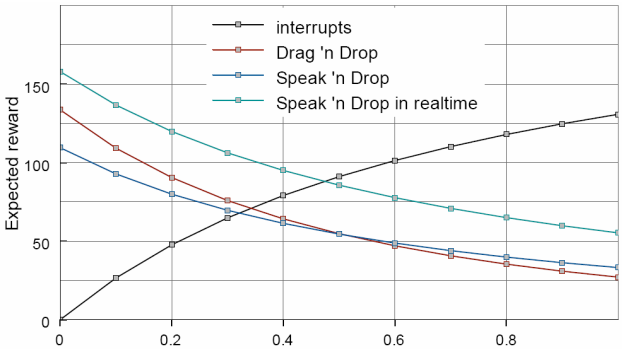

Figure 15. Predictive comparison of performance of Drag & Drop and Speak & Drop interaction techniques

For instance, Figure 15 (left-hand side of the diagram) shows that without interruptions the Speak and Drop interaction technique is less efficient than the Drag and Drop. However, the right-hand side of the diagram shows that this interaction technique becomes more efficient when the number of interrupt reaches a certain level (more than 1 interruption per minute which corresponds to the x-coordinate 0.5 of the diagram). In all circumstances, the Speak and Drop interaction techniques with a real time speech recognition engine performs far better that the others. Once the stochastic automata are composed in parallel as described in the previous sections, it is possible to perform many different types of analysis and observe the corresponding changing performance by varying the action rates given in Table 1. For example, the performance of the speech recognition system is governed by the variable "ee"; varying its value from 1000 to 120 results in the two curves labeled Speak'n'Drop and Speak'n'Drop in realtime of figure 15 respectively. With the same technique we have explored aspects concerning for example the different behavior of users with respect to motor skills and the effect of learning. The interested reader is referred to [14] for further details.

5. DISCUSSION AND CONCLUSION

Interruptions are unpredictable and quite often cannot be disregarded by the user in the working environment. As users are faced with increasing sources of information competing for their attention at any time, it is important to understand how interruptions affect our abilities to complete tasks. In the context of the design of safety critical interactive systems it is important to be able to predict the impact of interruptions on the performance of tasks.

In this paper we have proposed a design process for the design and assessment of interruption-tolerant systems. The feasibility of such an approach is illustrated by a case study that enables us to compare two interaction techniques with respect to their tolerance to interaction techniques with respect to interruptions. The presented case study was purposely minimalist so that we could highlight the approach without the dealing with the complexity of a case study. However, we are currently working on the application of the approach to dynamic reconfigurable systems [27]. Indeed, in many safety critical applications the dynamic reconfiguration is a main requirement to make the system more resilient to failures or malicious attacks. In such a context, we need appropriate analysis techniques to determine which would be best system configuration (which may include the choice of an interaction technique).

By injecting values, we can assess the performance under constraints we want to investigate. In order to better estimate the performance of task performance in safety critical systems, we could have combined the inject values extracted from the literature (i.e. ICS and Fitts law) with data collected from running applications which would include the mean time between failure, the mean time to recover from failure, availability of services, denial of service and so on. However, the accuracy of the values injected in the performance analysis is not the main issue in this proposal because the main aim is to provide an analytical tool to assess the impact of interruptions in task performance rather than provide accurate estimation of running applications. Moreover, the performance analysis we performed should be considered in a broader picture of the user experience and their results should be considered together with the perceived usability of the system. Therefore, this work should be considered as a first attempt towards a general design method for designing interrupt-tolerant Interactive Systems.

ACKNOWLEDGEMENT

The research in this paper is partly funded by EU network of excellence Resist/Faerus (IST-2006-026764).

REFERENCES

[1] Accot J. and Zhai S. (1997). Beyond Fitts' law: models for trajectory-based HCI tasks. Proc. of ACM CHI 97, 295–302

[2] Curzon, P., Ruksenas, R. & Blandford, A. (2007) An approach to formal verification of human-computer interaction. Formal Aspects of Computing. 513-550.

[3] Bailey, B.P. Konstan & J.A. Carlis, J.V. (2000). Measuring the effects of interruptions on task performance in the user interface. IEEE Int. Conf. on Systems, Man, and Cybernetics 2000. Vol. 2, 757-762.

[4] Barnard, P.J. and Teasdale, J.D. (1991). Interacting cognitive subsystems: A systemic approach to cognitive-affective interaction and change. Cognition and Emotion, 5, 1-39.

[5] ter Beek M., Faconti G., Massink M., Palanque P. & Winckler M. Resilience of interaction techniques to interrupts - A formal model-based approach. In proc. of INTERACT 2009, LNCS, Springer, 456-472.

[6] Bourgeois, F., Guiard, Y., & Beaudouin-Lafon, M. (2002). Multi-scale pointing: Facilitating pan-zoom coordination.

ACM SIGCHI Conf. on Human Factors in Computing Systems CHI 2002, 758-759. ACM Press.

[7] Cades D. M., Trafton J. G., Boehm-Davis D. A. & Monk C. A. (2007) Does the difficulty of an interruption affect our ability to resume? Proc. of the Human Factors and Ergonomics Society (HFES'07), 234-238

[8] Chisholm, C.D., Collison, E. K.,Nelson,D. R., & Cordell, W. H. (2000). Emergency department workplace interruptions: Are emergency physicians "interrupt-driven" and "multitasking"? Academic Emergency Medicine, 7, 1239–1243.

[9] Card, S.K.; T.P. Thomas & A. Newell (1983), written at London, The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, ISBN 0898592437.

[10] Li, S., Blandford, A., Cairns, P. & Young, R. M. (2008). The Effect of Interruptions on Post-Completion and Other Procedural Errors: An Account Based on the Activation-Based Goal Memory Model. Journal of Experimental Psychology: Applied. 14.4. 314-328.

[11] Czerwinski, M., Horvitz, E., & Wilhite, S. (2004). A diary study of task switching and interruptions. In Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems CHI '04. ACM Press, 175-182.

[12] Diaper, D., Stanton, N. A. (eds.) (2004). The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, 2004. 650 pages.

[13] Diez M., Boehm-Davis D. A. & Holt R. W. (2002) Model-based predictions of interrupted checklists, in: Proc. of the 46th M. of Human Factors and Ergonomics Society, 250-254.

[14] M.H. ter Beek, G.P. Faconti, M. Massink, P. Palanque and M. Winckler (2009), Resilience of Interaction Techniques to Interrupts: A Formal Model-based Approach - Full Version. Technical Report 2009-TR-001, ISTI-CNR.

[15] Fitts, Paul M. (1954): The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. In Journal of Experimental Psychology, 47 pp. 381-391.

[16] Gillie T. & Broadbent D. (1989). What makes interruptions disruptive? A study of length, similarity and complexity, Psychological Research, 50 (4), 243-250.

[17] Hopp, P. J, Smith, C. A. P., Clegg, B. A. & Heggestad, E.D. (2005). Interruption Management: Use of Attention-Directing Tactile Cues. Human Factors, Vol. 47, No.1, pp.1–11.

[18] Horvitz, E. & Apacible, J. (2003). Learning and reasoning about interruption. In Proc. of 5th Int. Conf. on Multimodal interfaces. ICMI '03. ACM Press, 20-27.

[19] Iqbal, S. T. & Horvitz, E. (2007). Disruption and Recovery of Computing Tasks: Field Study, Analysis, and Directions. In Proc. of ACM CHI 2007, ACM Press, 677-688.

[20] Iqbal, S. T. and Bailey, B. P. 2008. Effects of intelligent notification management on users and their tasks. In Proc. of ACM CHI '08. ACM Press, 93-102.

[21] Jambon F. (1996) Formal modelling of task interruptions. In Proc. of ACM SIGCHI conf. CHI'96, Conference Companion. ACM Press, 45-46.

[22] McCrickard D. S. & Chewar C. M. (2003) Attuning notification design to user goals and attention costs, Communications of ACM, 46 (3), 67-72.

[23] McFarlane D. C. (1999) Coordinating the interruption of people in human-computer interaction. Proceedings of INTERACT'99, Amsterdam: IOS Press, 295-303.

[24] Monk, C., Boehm-Davis,D., & Trafton, J. G. (2004). Recovering from interruptions: Implications for driver distraction research. Human Factors, 46, 650–663.

[25] Morris, D., Brush, A. B., and Meyers, B. R. (2008). SuperBreak: using interactivity to enhance ergonomic typing breaks. In Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems. CHI '08. ACM, Press, 1817-1826.

[26] National Transportation Safety Board. (1988). Aircraft accident report: Northwest Airlines, Inc., McDonnell Douglas DC-9-82, N312RC, Detroit Metropolitan Wayne County Airport, Romulus, Michigan, August 16, 1987 (NTSB/AAR-88/05).Washington, DC.

[27] Navarre, D., Palanque, P. & Basnyat, S., (2008) Usability Service Continuation through Reconfiguration of Input and Output Devices in Safety Critical Interactive Systems. The 27th Int. Conference on Computer Safety, Reliability and Security (Safecomp 2008), Springer LNCS 5219, pp. 373–386.

[28] O'Conaill B. & Frohlich D. (1995) Timespace in the workplace: Dealing with interruptions, in: Human Factors in Comp. Systems: CHI'95 Companion, ACM Press, 262-263

[29] Oulasvirta, A. and Saariluoma, P. (2006). Surviving task interruptions: Investigating the implications of long-term working memory theory. Int. J. Hum.-Comput. Stud. 64, 10, 941-961.

[30] Peterson J. (1981), Petri Net Theory and the Modeling of Systems, Prentice Hall.

[31] Palanque P. & Bastide R. (1995). Verification of an Interactive Software by analysis of its formal specification. Proc. of the IFIP TC 13 Interact'95, p. 191-197.

[32] Palanque, P., Bastide, R. (1997). Synergistic modeling of tasks, system and users using formal specification techniques. Interacting With Computers, Academic Press, 9, 12, 129-153

[33] Paterno, F., Mancini, C. & Meniconi, S. (1997). ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proc. of Interact'97. Chapman & Hall, 362-369.

[34] Speier C., Vessey I. & Valacich J. S. (2003) The effects of interruptions, task complexity, and information presentation on computer-supported decision-making performance, Decision Sciences, 34 (4), 771-797.

[35] Trafton, J. G., & Monk, C. A. (2007). Task Interruptions. Reviews of Human Factors and Ergonomics, 3, 111-126.

[36] West, R. L., Nagy, G. (2007) Using GOMS for Modeling Routine Tasks Within Complex Sociotechnical Systems: Connecting Macrocognitive Models to Microcognition. Journal of Cog. Eng. and Decision Making, Vol. 1, n° 2, pp. 186-211