

TASK PATTERNS FOR TAKING INTO ACCOUNT IN AN EFFICIENT AND SYSTEMATIC WAY BOTH STANDARD AND ERRONEOUS USER BEHAVIOURS

Philippe Palanque & Sandra Basnyat

LIHHS-IRIT, University of Toulouse 3, 118, route de Narbonne, 31062 Toulouse cedex, France

Abstract: While designing interactive software, the use of a formal specification technique is of great help because it provides non-ambiguous, complete and concise notations. The advantages of using such a formalism is widened if it is provided by formal analysis techniques that allow to prove properties about the design, thus giving an early verification to the designer before the application is actually implemented. However, formal specification of interactive systems (even though aiming to produce reliable software) often does not address the issues of erroneous user behaviour. This paper tackles the problem by proposing a systematic way of dealing with erroneous user behaviour. We propose to extend task models (describing standard user behaviour) with erroneous user behaviour. Without appropriate support, incorporating erroneous user behaviour in task models requires much effort from task analysts. We thus propose the definition of patterns of user errors in task models. These patterns of errors are then systematically applied to a task model in order to build a task model covering both standard and erroneous user behaviour. These task models can then be exploited towards system models to provide a systematic way of assessing both system compliance to user tasks and system tolerance to user errors.

Key words: Formal specification, human error, UI design, Petri nets, tasks models

1. INTRODUCTION

While designing interactive software, the use of a formal description technique is of great help by providing non-ambiguous, complete and concise notations. The advantages of using such a notation is widened if it is provided by formal analysis techniques that allow proving properties about the design, thus giving an early verification to the designer before the application is actually implemented. However, formal description of an interactive system (even though aiming at producing reliable interactive software) often do not address the issue of erroneous user behaviour. Indeed, the focus is mainly on describing ‘normal’ behaviour of users while difficulties mainly arise due to unexpected or invalid (according to the system’s description of valid behaviour) user actions performed.

This paper reports work done in combining issues relating to formal description techniques for interactive systems and issues raised by techniques for human error analysis and categorisation. The basic idea is to bring together in a unifying framework, three aspects of the user-centred development process for reliable interactive software i.e. task analysis and modelling, formal description of the system and human error categorisation.

Task analysis and modelling approaches have always focussed on standard behaviour of users leaving user error analysis for later phases in the design processes (Barber & Stanton 2004). This is part of the rationale underlying task analysis which is to provide an *exhaustive* analysis of user behaviour. This comprehensivity (of analysis) is critical as it is meant to provide the basics for a global understanding of user behaviour and tasks that will serve as a basis for driving evolutions of the interactive system. However, practice shows that reaching this exhaustivity is very difficult in terms of availability of resources and economy. These aspects drive people responsible for task analysis and modelling to focus on most frequent and standard activities, thus leaving the infrequent or erroneous ones unconsidered. However, this is precisely where the emphasis should be placed in order to deal efficiently with error tolerance.

In this paper we propose to use task patterns as a way of dealing exhaustively with potential user errors. These patterns of tasks have been modelled and can be directly reused within a task model in order to represent potential deviations of user behaviour. These task models (including representations of possible user errors) can then be tested over a system model in order to verify that the system under development is able to tolerate such erroneous user behaviour. The principle is similar to the work presented in (Fields et al., 1999) in terms of objective. However, in this paper we focus not only on ways of identifying erroneous user behaviour, but also on

representing such behaviour and on integrating them with standard user behaviour.

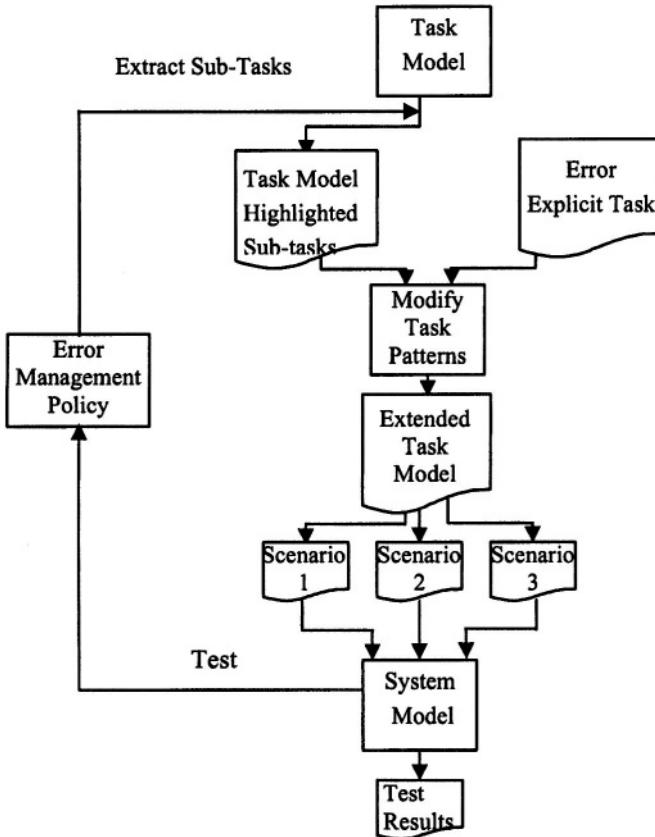


Figure 1. Overview of the research

The paper is structured as follows. Section 2 presents the domain of task analysis and modelling. It presents related work in this field and introduces the notion of task patterns as shown above. Section 3 deals with human error issues. It presents a subset of an extensive classification we have defined in order to identify the set of possible erroneous user behaviour that could occur while interacting with an interactive system. The classification builds upon previous work in the field of human error analysis and is then combined with the task patterns presented in section 2. Section 4 shows, on a case study, how this framework can be used and what it brings to the design and verification of error-tolerant safety critical interactive systems. Section 5 briefly presents some ways for relating this work on task modelling to work on the system side modelling.

2. TASK MODELLING

Tasks analysis and modelling is a central element to user centred design approaches. For this reason a lot of work has been devoted to it and to its integration in the development process of interactive systems. This section first presents an informal presentation of tasks modelling. A summary of task modelling techniques is then provided followed by a third section dealing with previous work addressing the combination of user error and task modelling. Finally, we present the notion of patterns and how this notation can be applied to task modelling.

2.1 Informal Presentation

A task model is a representation of user tasks often involving some form of interaction with a system influenced by its contextual environment. We use the word “influenced” (as opposed to “driven” by the environment) to highlight our thoughts on user’s having an underlying goal and hence plan in their mind before attempting to perform a task. This contrasts Suchman’s 1987 proposal that plans are representations of situated actions produced in the course of action and therefore they become resources for the work rather than they in any strong sense determine its course.

Users perform tasks, which are structured sets of activities (Preece 1994) in order to achieve higher-level goals. Tasks can be further decomposed corresponding to lower level sub goals. This notion of decomposition naturally results in tree-like structures and thus hierarchical representation of the model. More recently, task modelling tools such as Paternò’s ConcurrentTaskTrees CTT (Paternò 1999) have enabled the distinction of abstract, user, interaction and application tasks as well as the possibility to model temporal aspects of activities when specifying a model. The CTT notation is described further in this section. The typical characteristics of a task model include its hierarchical structure, the task decomposition and sometimes the temporal relationship between elements.

Table 1. Summary of Task Analysis Techniques





Acronym	Full Name
HTA	Hierarchical Task Analysis (Annett & Duncan, 1967)
TKS	Task Knowledge Structure (Johnson, 1989)
MAD	Méthode Analytique de Description de tâches (Scapin et al., 1989)
UAN	User Action Notation (Hix & Hartson 1993)
GTA	GroupWare Task Analysis (van der Veer, 1996)
CTT	ConcurTaskTrees (Paternò, 1999)
GOMS	Goals, Operators, Methods and Selection rules (Baumeister, 2000)

The above mentioned models differ in their type of syntax (textual vs graphical), the level of formality and the richness of operators offered to designers. The work presented here could fit any of these notations even though our focus is on CTT (because of its tool support). More recent approaches to task modelling have emphasised the context in which the interaction between human and system is taking place with respect to user characteristics, organisations, artefacts to be manipulated and actions required. This has resulted in many task models having hybrid conceptual frameworks.

2.2 CTT and why it is not enough

ConcurTaskTrees (CTT) is a graphical notation used for specifying task models of cooperative applications in a hierarchical structure while also denoting temporal operators. The task models can then be simulated to study different possible paths of interaction. The notation is based upon four types of tasks, abstract tasks, user tasks, application tasks and interaction tasks as shown in Table 2.

Table 2. CTT Tasks

Graphical Symbols	Description
	Abstract Tasks: Tasks that require complex activities whose performance cannot be univocally allocated.
	User Tasks: Usually they are important cognitive activities.
	Application Tasks: Can supply information to the user.
	Interaction Tasks: Between the user and the system.

The operators provided within the CTT notation are described in Table 3. Figure 2 depicts the “choice” operator in which the interactive “Task” is performed by either interactive task 1 or interactive task 2. Figure 2 illustrates the use of an “abstract” task which is performed firstly by a user task, perhaps cognitive, which enables the interactive task.

Figure 4 is a simple example using the CTT notation taken from the ConcurTaskTree Environment (CTTe) tool for accessing an ATM with emphasis and expansion on the withdraw cash task.

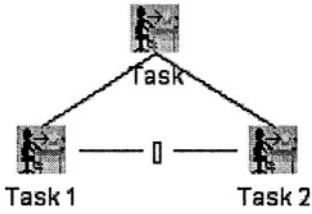


Figure 2. Choice Operator

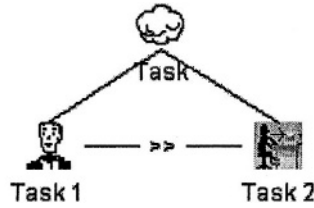


Figure 3. Enabling Operator with varied task type

Table 3. Operators used in the CTT notation

Notation	Description
T1 >> T2	Enabling
T1 []>> T2	Enabling with information processing
T1 > T2	Deactivation
T1 [] T2	Choice
T1 *	Iteration
T1 [I] T2	Concurrency with information exchange
T1 > T2	Suspend resume
T1 T2	Independent concurrency
T1 (n)	Finite iteration
[T1]	Optional task

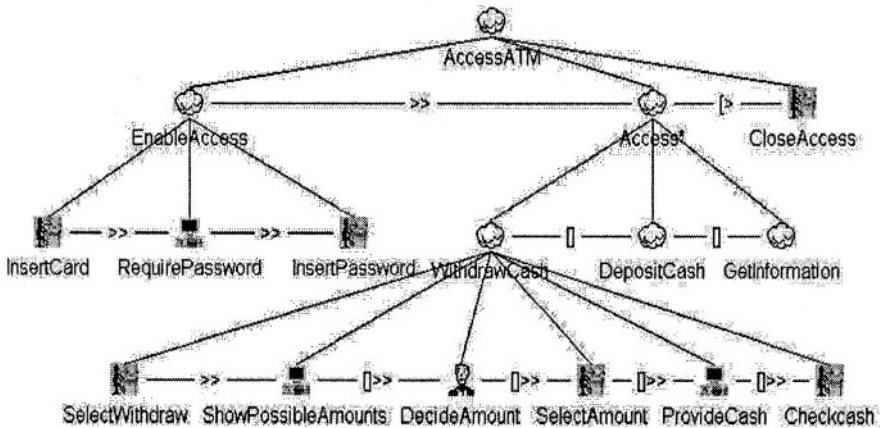


Figure 4. CTT ATM Example provided within CTTe

Figure 4 shows, at a level of description of most task analysis methods, with addition of temporal operators, the task of accessing an ATM. The diagram is read from left to right. A task is not complete until its necessary sub-tasks are addressed. Thus in the above example, EnableAccess must

first be completed by performing of its subtasks. That is, the user inserts the card (an interactive task) which enables the request of the PIN (a system task) which in turn enables the entering of the PIN (a interactive user task). The success of the EnableAccess task enables the Access task which is composed of WithdrawCash or DepositCash or GetInformation. Please note however, that the Access task is iterative which means that each of the Access subtasks can be performed in any order. The iteration of the Access task is aborted by the interruption of the CloseAccess task.

We are considering CTT as the most appropriate notation for this research because of its graphical appearance and tool support.

2.2.1 Why CTT is not enough

There are also downsides to the CTT notation. The context and environmental conditions within which the activities are taking place are not considered when modelling tasks in CTT. Surrounding circumstances could have effects on the process. Also, details of artefacts being manipulated during tasks could be useful because the time spent performing a task could be dependent on the artefact in hand. The cognitive workload and users' current state is also not detailed. For example, conditions such as stress or tiredness or being under pressure may effect the way in which actions are performed with respect to their efficiency and effectiveness. CTT does not allow the designer to detail the type of low-level interaction taking place such as a mouse click or a keyboard entry. Furthermore, the "shift of focus" of the user when interacting with an interface is not taken into account. This level of detail could be crucial in safety-critical interactive systems.

2.3 Task modelling and user errors

As stated in the introduction, when modelling user behaviour, an error-free perspective is usually employed. It is normally during the testing phase of the system development cycle that errors are realised and taken into account. Task modelling as yet, does not allow for the description, representation and analysis of unexpected eventualities that may occur including human error. Since the task model influences the system design it is important to understand how to manage and overcome possible errors.

In their paper, Baber and Stanton (Barber & Stanton 2004) propose that a system image implies a set of routines that a person may use and that the selected routine will be based on the user's goal and previous experiences. They suggest a need to represent interaction between user and product to consider possible mental model mismatches between a given system image

and user representation during initial design activity. The proposed technique, Task Analysis for Error Identification (TAFEI) is based on the assumption that interaction is goal-oriented and passes through a series of states. The TAFEI approach consists of three stages, a Hierarchical Task Analysis (although any technique is acceptable), construction of a State Space Diagram (SSD) mapped with the HTA plans, finally construction of a transition matrix to display state transitions during device use. This work tries to address the same goal as ours. However, there are three main differences:

- Our work, exploiting classification of user error provides a systematic way of dealing with possible user errors,
- Our proposal to define and exploit task patterns for user errors provides a way of coping in an efficient and reliable way with the complexity of the task models,
- The authors consider as erroneous only the paths in a users task that are provided by the system but do not support the achievement of the user's goal. We consider user error in a broader sense including errors such as mistakes and slips.

Paternò and Santoro (2002) also suggest that a goal is a desired modification of the state of an application. Their work describes how task models can be used in an inspection based usability evaluation for interactive safety-critical applications.

A key aim to their method is to provide designers with help in order to systematically analyse what happens if there are deviations in task performance with respect to what was originally planned during the system design. Building upon the HAZAOP family of techniques a set of predefined classes of deviations are identified by guidewords such as “none”, “other than” and “ill-timed”.

The method incorporates three stages:

- 1) Development of the task model for the application considered.
- 2) Analysis of deviations related to the basic tasks.
- 3) Analysis of deviations for high-level tasks.

The analysis, which is recommended be carried out by interdisciplinary groups, follows a bottom-up approach considering basic and then high-level tasks. Documentation of the analysis takes the form of tables. As we suggest in this paper, (Paternò & Santoro 2002) mention that the interpretation of every guideword for every task of the prototype addresses completeness issues however has the drawback of taking time and effort.

This work tries to address the same goal as ours. However, there are two main differences:

- Our proposal to define and exploit task patterns for user errors provides a way of coping in an efficient and reliable way with the complexity of the task models,
- The authors consider error analysis based only on a set of high-level guidewords (such as “other-than”) and thus leaving detailed analysis to the analyst’s discretion. Our proposal is much more concrete and grounded on previous work in the field of user error identification and classification. For instance we have more than 20 types of errors that would fit within the “other-than” guideword (for instance “branching error”, “environmental capture”, “order error”...).

The Technique for Human Error Assessment or THEA (Pocock et al., 2001) is aimed at helping designers of interactive systems anticipate human errors resulting in interaction failures. As we suggest, this is also a technique intended for use during the initial phases of the development lifecycle. With its foundations laying in human-reliability analysis (HRA) (Kirwan 1994) it aims to establish requirements for “error resilient” system design. In their paper, it is noted that errors can be regarded as failures in cognitive processing (Pocock et al., 2001). The process of analysing a system’s vulnerability to human error is performed by posing provided questions about a given scenario, identifying possible causal factors of potential problems identified and finally identifying consequences and their impact on task, work, user, system, etc. The results are recorded in tables for further analysis.

The THEA approach shares a common perspective with our work, that is, we aim to assist designers to produce usable error tolerant interactive systems based on systematic techniques. However, our work differs in that we intend the task patterns to be applicable to more than one system design, possibly of various domains. This means that once a sound solution has been modelled, there will be less repetition of work.

2.4 Task patterns

Task patterns for interactive systems design is a relatively new concept with the aim of solving design problems using existing knowledge of previously identified patterns and solutions. The majority of research to date focuses on user interaction with software and interfaces providing interface design patterns and task based patterns to improve usability rather than task

based patterns that focus on user behaviour intended for testing the compatibility with system design.

Indeed, the focus is mostly on what we could call generic user behaviours. This work proposes then a set of user interface design solutions to such user behaviours. Work presented in (Sinnig et al., 2003) focuses on establishing and integrating patterns as building blocks for the creation of the task model in order to merge software and usability engineering. Work presented in (van Welie et al., 2000) also addresses the issues raised by user interaction stating that patterns for user interface design should help make systems more usable for humans.

Task patterns were first introduced by Paternò (Breedvelt et al., 1997 & Paternò 1999) as reusable structures for task models. The patterns were described as hierarchical structured task fragments that can be reused to successively build the task model.

(Sinnig et al., 2004) present a model-based approach to development in which they refer to the task, user, business, object, dialogue, presentation and layout models. In (Sinnig et al., 2003) they describe two types of patterns: task and feature. Task, which detail activities the user has to perform while pursuing a certain goal and feature patterns that are applied to the user-task model describing the activities the user has to perform using a particular feature of the system. Task patterns are said to be composed of sub-patterns which can be task or feature patterns. A four-stage strategy for the process of pattern application is also detailed. Steps include Identification, Selection, Adaptation and Integration.

Such task patterns are context specific and problem centred as opposed to guidelines which can often be too simplistic, too abstract and difficult to interpret. The task patterns proposed in these works only model error-free behaviour, that is, the possibility of human-error would be considered later in the design lifecycle during the testing phase of the system for example.

3. USER ERRORS

Human error plays a major role in the occurrence of accidents in safety-critical systems such as in aviation, railways systems, or nuclear power plants (Reason 1990). The recent rail disaster of North Korea has been officially blamed by their Government on human error. (BBC News, 2004).

Interactive systems particularly those that are safety-critical need to be designed with the eventuality of human error in mind to prevent catastrophes. This means, early in the design process and as well as during the testing phase. Although the term “human error” appears very

controversial, theories of human errors such as Rasmussen's (Rasmussen 1983) SRK, Hollnagel's (Hollnagel 1991) Phenotypes and Genotypes and Norman's (Norman 1988) classification of slips can be considered widely acceptable.

Table 4 presents a subset of the Skill-based Human Error Reference Tables (we will refer to as HERT from this point onwards). Within Skill-based errors are two main failure modes, inattention and over attention (Reason 1990). This table details those errors found within the inattention failure mode.

3.1 Classification of user errors

Hollnagel's (Hollnagel 1991) error classification scheme is an example of a behaviour-based taxonomy of human error (Reason 1990). Hollnagel identifies eight simple phenotypes and five complex phenotypes. It starts with the observable phenomena, such as errors of omission or commission, rather than cognitive theories. The observable phenomena, he states, make up the empirical basis for error classification. He refers to these behavioural descriptions as the *phenotype* of human error. The error *genotype* denotes the mental mechanism assumed to underlie the observable erroneous behaviour.

Rasmussen (Rasmussen 1983) proposed the SRK theory in which he distinguished three levels of human processing each with its associated error types: (1) skill based level, for activities performed automatically, (2) rule based level, for circumstances in which our intuition provides an applicable response, and (3) the knowledge based level, for new situations in which there are no rules.

Based on Rasmussen's SRK theory, Reason developed the Generic Error Modelling System (GEMS) (Reason 1990) which can be summarised as:

- SB: Unintended deviations from the procedures that are conducted automatically by the individual.
- RB: Associated with those activities in which the individual has to consciously choose between alternative courses of action.
- KB: The individual attempts to define a new procedure on the basis of knowledge about the system they are using.
-

Using the above-mentioned classifications, we have produced detailed tables, grouping together many user oriented error types appropriate for our studies based fundamentally on the SRK theory.

The tables are decomposed at the following levels: (Please note part of sections 1.3, 1.4 and 1.5 are shown in Table 4)

- 1 Rasmussen's Skill-based level
 - 1.1 Rasmussen's SRK
 - 1.2 Reason's failure modes (inattention & over attention)
 - 1.3 Reason's 6 common mechanisms
 - 1.4 Error name & definition
 - 1.5 Example of error
- 2 Rasmussen's Rule-based level
 - 2.1 Reason's failure modes (misapplication of good rules & application of bad rules)
 - 2.2 Reason's 9 common mechanisms
 - 2.3 Error name & definition
 - 2.4 Example of error
- 3 Rasmussen's Knowledge-based level
 - 3.1 Reason's failure modes (selectivity, workspace limitations etc)
 - 3.2 Error name & definition
 - 3.3 Example of error

Some of the examples identified in the HERTs are those presented by authors of the classifications, others have been supplemented. Within the framework, further classifications are identified such as Hollnagel's eight simple phenotypes and five complex phenotypes (Hollnagel 1991), Norman's six categories of slips (Norman 1988), HAZOP causes of deviations (HAZOP 1989) etc., which were not necessarily located together within the tables. The benefit of producing such reference tables enables the exact identification of very precise error types when analysing human behaviour associated to particular tasks of a task model.

3.2 Relating types of user error to task patterns

Since the HERTS have been decomposed to the level of an example for each type of error, it is possible to relate every classified error to a particular task. Thus for each task of a task analysis model, it is possible to determine, by means of elimination, which human errors are applicable.

Once specific error-related situations for each task of a task model have been identified, it is possible to re-design the model to support and make explicit these types of errors. At this stage, if there is an existing system in place, its constraints will need to be considered. This ensures, that when the system or future system is modelled, for example as a Petri-net, the task model can be tested over the system model with the human errors already

identified. This would avoid the discovery of problems during later testing phases of the development lifecycle.

Finally, the re-designed error explicit parts of the task model can be ‘plugged in’ to the relevant areas of the task model creating task patterns which can be applied to other domains involving the same or similar activities.

Table 4. Subset of the Skill-Based Inattention Human Error Reference Tables

Reason's 6 Common Mechanisms	Error Name & Definition	Examples of errors	Error Type Ref
Double-capture slips Involve two distinct, though casually related, kinds of capture. (Reason 1990) (Norman's 1989 1st category of slips)	Strong-habit intrusion The unintended activation of the strongest action scheme beyond the choice point. (Reason 1990) Hollnagel 1993, simple phenotype, Intrusion.	Unintended activation of the strongest action schema beyond the choice point. (Reason 1990)	Reason 1990
	Error mode: action not included in current plans.	On starting a letter to a friend, I headed the paper with my previous home address instead of my new one. (Reason 1990)	
	Corresponding complex phenotype, Hollnagel, 1993 capture, branching and overshoot.		
	Strong-habit capture (Reason 1990)	I intended to stop on the way to work to buy some shoes, but ‘woke up’ to find that I had driven right past. (Reason 1990)	Reason 1990

4. CASE STUDY (CASH MACHINE)

In order to demonstrate our ideas, we take the example of an Automated Teller Machine (ATM). ATMs have often been used to demonstrate task analysis since they are widely used systems demonstrating clear human-computer interaction. We are using the ATM as an example because we are focusing on repetitive, highly structured, situated based systems involving less decision making, however errors can still occur.

4.1 Task Models

Since the possible interactions with an ATM may sometimes be complex and plentiful, we have focused our attention on producing a task model for the process of withdrawing cash using the CTT notation (See Figure 5).

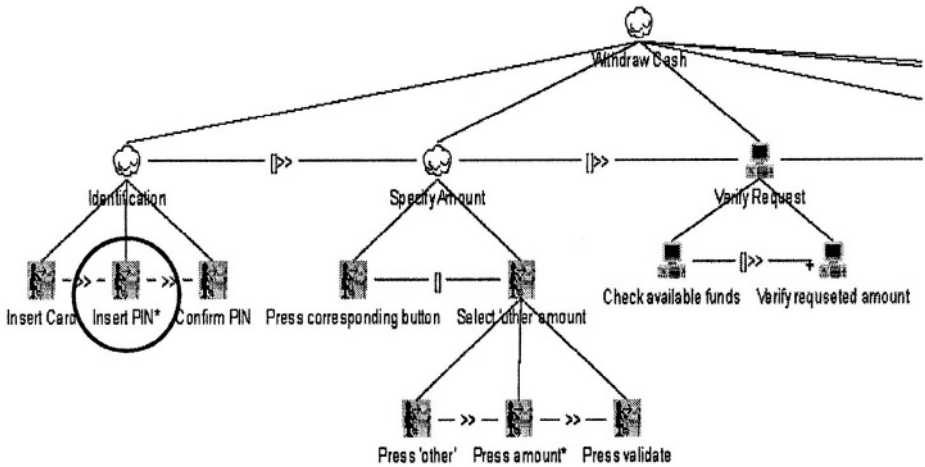


Figure 5. Sub-section of CTT Task Analysis for Withdrawing Cash at an ATM (Highlighting the “Insert PIN” activity for analysis during the case study)

The CTT diagram for cash withdrawal at an ATM shows part of the process of being identified, selecting an amount through to the withdrawal of the money. We are referring to this CTT as a general representation of task analysis because it describes error free behaviour and “normal” flow of activities. For example, if the same task was analysed using the HTA approach, the resulting textual or graphical representation would be somewhat similar. However CTT allows for temporal operators to be described, which is particularly useful for modelling interaction with an ATM, resulting in a more realistic model in terms of possible user activities.

4.2 Identification of possible deviations

Following the task analysis and modelling, each low-level sub-task can be considered for possible human error events that may occur. This can be done by referring to the HERTs and determining whether or not each particular type of error could occur to each CTT task. In this paper, we present the systematic analysis of possible human deviations while interacting with an ATM based on the “insert PIN” sub-task of Identification highlighted in Figure 5 and the Skill-Based HERTs (a subset of which is

shown Table 4). Where applicable, examples of human deviation relevant to the Insert PIN activity have been identified. This can be seen in Figure 6.

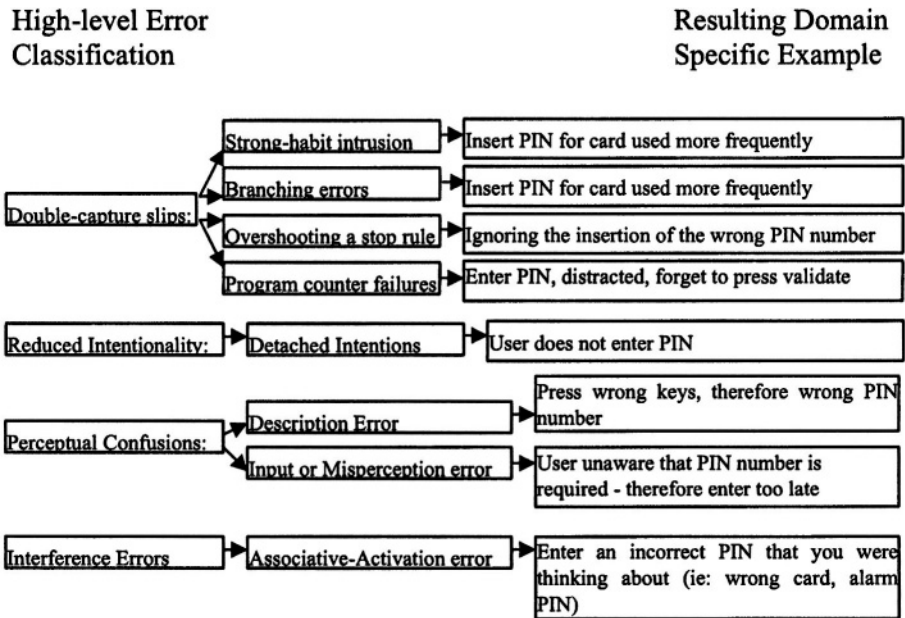


Figure 6. Subset of the systematic analysis of possible deviations while inserting a pin at an ATM based on a CTT task analysis model and skill-based human error classification for the inattention failure mode

It must be noted however, that although many possibilities have been considered, we have sensibly limited our analysis to avoid considering natural disasters such as “dropping down dead” at the cash machine.

4.3 Including task patterns for errors

Some of the errors identified in Figure 6 above can be summarised as they result in the same task modelling. For example, ‘Insert PIN for card used more frequently’ and ‘Press wrong keys, therefore wrong PIN number’ can both be considered as entering the wrong PIN. Figure 7 to Figure 17 illustrate the re-designed task models for the Insert PIN sub-task which make explicit the possible deviations. The task models contain repetitions which can now be re-labelled as task patterns.

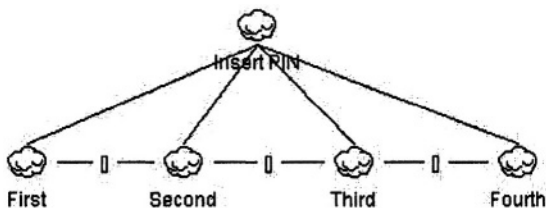


Figure 7. Four possibilities of interaction when entering PIN

Taking into account the potential errors identified, there are four possibilities of interaction with the system when entering the PIN shown as four abstract tasks, first, second, third and fourth in Figure 7. The abstract tasks are made up of combinations of five task patterns labelled P1, P2, P3, P4 and P5. These patterns can be seen in Figures 8 – 12.

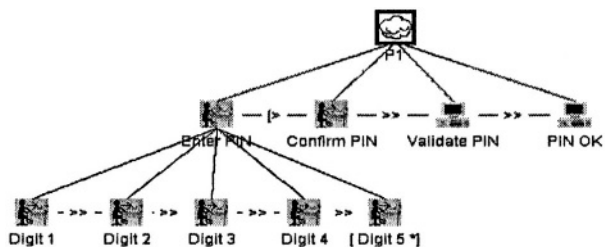


Figure 8. Pattern 1: PIN OK

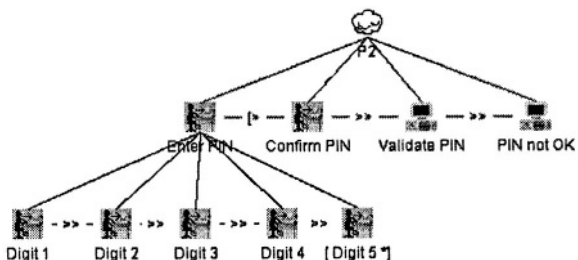


Figure 9. Pattern 2: PIN not OK

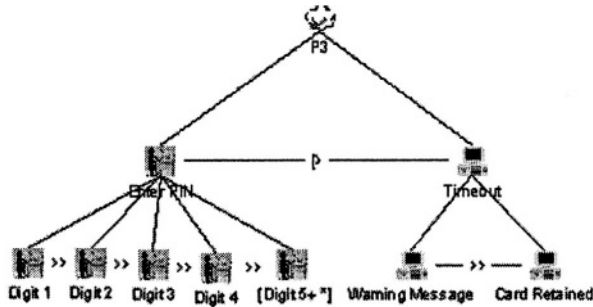


Figure 10. Pattern 3: Too long entering PIN

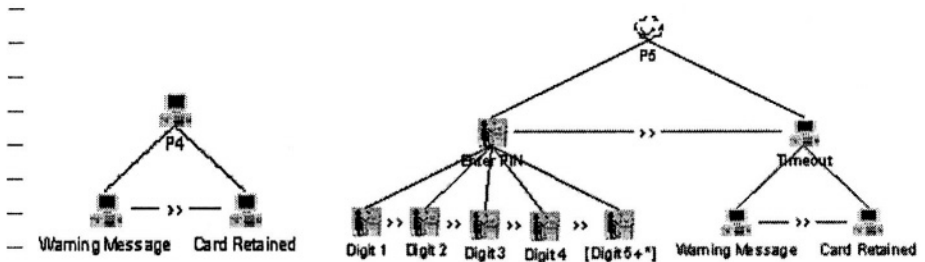


Figure 11. Pattern 4: Simple Timeout

Figure 12. Pattern 5: Timeout on PIN confirmation

1) For the first possibility of interaction (refer to Figure 7), either:

- The PIN is correct on the 1st try OR
- The user takes too long to enter the PIN and the system times out OR
- The user does not enter a PIN at all and the system times out OR
- The user takes too long confirming the PIN and the system times out

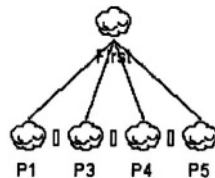


Figure 13. [First possible interaction]

2) For the second possibility of interaction (refer to Figure 7), either:

- The PIN is incorrect on the 1st try and correct on the 2nd try OR
- The PIN is incorrect on the 1st try and the user takes too long to enter the PIN on the 2nd try therefore the system times out OR
- The PIN is incorrect on the 1st try and the user does not enter a PIN at all on the 2nd try and therefore system times out OR
- The PIN is incorrect on the 1st try and the user takes too long confirming the PIN on the 2nd try and the system times out

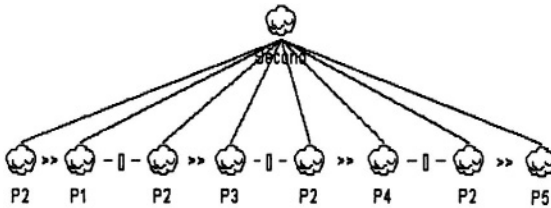


Figure 14. Second possible interaction

- 4) For the third possibility of interaction (refer to Figure 7), either:
- The PIN is incorrect on the 1st try, incorrect on the 2nd try and correct on the 3rd try OR
 - The PIN is incorrect on the 1st try, incorrect on the 2nd try and the user takes too long to enter the PIN on the 3rd try therefore the system times out OR
 - The PIN is incorrect on the 1st try, incorrect on the 2nd try and the user does not enter a PIN at all on the 3rd try and therefore system times out OR
 - The PIN is incorrect on the 1st try, incorrect on the 2nd try and the user takes too long confirming the PIN on the 3rd try and the system times out.

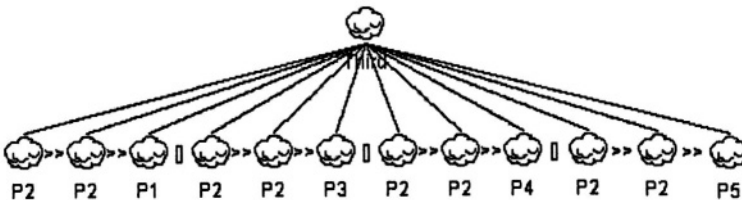


Figure 15. Third possible interaction

- 5) For the fourth possibility of interaction (refer to Figure 7):
 The PIN is incorrect on the 1st try, incorrect on the 2nd try and incorrect on the 3rd try

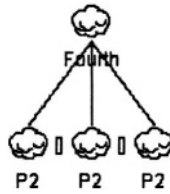


Figure 16. Fourth possible interaction

It can be seen from previous models that, taking into account possible erroneous behaviour increases significantly the size of the task models. For instance there are 21 leaves in the full task model for normal user behaviour (some of which is represented in Figure 5) and 234 leaves in the task model describing both erroneous and normal user behaviour (merging of models illustrated in Figures 7 to 17).

5. TOWARDS ERROR-TOLERANT SYSTEMS

This paper has presented the process of transforming a simple task analysis model to a more complex one making domain specific human errors explicit by means of task patterns that could be applied to other domains.

It is intended that the idea of identifying human errors early in the development process will enable the design of error-tolerant systems. We have previously studied different ways of taking into account task models in interactive systems development.

For space reasons, we only present in this section some ongoing work we are carrying out to exploit the results presented in this paper.

The process of relating task models and system models extends previous work presented in (Navarre et al., 2001) where task models are used in combination with system models in order to verify that both models were compliant with each other.

In this work we were only considering error-free user behaviour and were able to prove compatibility of tasks and systems at lexical, syntactic and semantic levels. More information about that can be found in (Palanque et al., 1997) and in (Palanque & Bastide 1997). Such verification allows designers to assess that all the tasks in the task model correspond to actions offered by the system (and represented in the system model). Similarly, the sequence of tasks in the task model must be compatible with the valid sequence in the system model.

6. CONCLUSION

In this paper we have presented a way of taking into account in a systematic way erroneous user behaviour. This work builds upon previous work in the field of task analysis, task modelling, human error analysis and identification. We have proposed the definition and use of task patterns for dealing with complexity and repetitions that frequently appear when modelling erroneous user behaviours. We have shown on a simple case study how task patterns have been identified and how we have modelled them using CTT notation and its related tool CTTE. Due to their intrinsic nature, patterns are good candidates for known and previously encountered problems. This is the reason why they have been successfully exploited on the ATM case study. Their application in other context where interaction techniques are more innovative has still to be studied. In the same way as we have studied the use of patterns for CUA interactors, we are currently working on their application for ARINC 661 user interface standard in cockpit displays in order to provide certification authorities in France DPAC (Direction des Programmes de l'Aviation Civile) with systematic error identification techniques.

ACKNOWLEDGMENTS

The work presented in the paper is partly funded by French DGA under contract #00.70.624.00.470.75.96 and EU via the ADVISES Research Training Network RTN2-2001-00053.

REFERENCES

- Alexander, C., Ishikawa, S and Silverstein, M. (1977) A pattern language: towns, buildings, construction.
- Annett, J. and Duncan, K., (1967) Task Analysis and Training Design, *Occupational Psychology*, 41,1967, pp.211-227.
- ARINC 661. (2002) Cockpit display system interfaces to user systems. Arinc specification 661. Published: April 22, 2002. An ARINC document prepared by airlines electronic engineering committee Published by aeronautical radio, inc. 2551 riva road, annapolis, maryland 21401
- Baber, C., and Stanton, N. (2004). Task Analysis for Error Identification. In D. Diaper & N. Stanton (Eds.) *The Handbook of Task Analysis for Human-Computer Interaction*. New Jersey: Lawrence Erlbaum Associates p.367-379

- Baumeister, L.K., John, B.E., Byrne, M.D. (2000). A Comparison of Tools for Building GOMS Models Tools for Design. In: Proc. of ACM Conf. on Human Factors in Computing Systems CHI'2000, ACM Press, New York, 502-509
- BBC News Website (2004) <http://news.bbc.co.uk/2/hi/asia-pacific/3656853.stm> (last accessed 30th April 2004)
- Blandford A. (2000). Designing to avoid post-completion errors. PUMA working paper WP33. (<http://www.cs.mdx.ac.uk/puma/>).
- Blandford, A., (2000) PUMA Footprints: linking theory and craft skill in usability evaluation. (<http://www.cs.mdx.ac.uk/puma/>).PUMA working paper WP26.
- Blandford, A. (2000) Designing to avoid post-completion errors. WP33
- Breedvelt, I., Paternò, F. & Sereriins, C. (1997). Reusable Structures in Task Models, Proceedings DSVIS '97, Springer Verlag, pp.251-265
- Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. (1996) Pattern Oriented Software Architecture. John Wiley & Sons, Inc.
- Card, S.K., Moran, T.P., Newell, A. (1983). The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale
- Farenc C. & Palanque P. (1999). A Generic Framework based on based on Ergonomic Rules for Computer Aided Design of User Interface. J. Vanderdonckt, A. Puerta (eds.), Computer-Aided Design of User Interfaces II, Proceedings of CADUI'99, Kluwer Academics, Dordrecht, 1999.
- Feldbrudge F., Jensen K. (1986). Petri net tool overview in Brauer W. (Editor). LNCS 254 & 255, Springer Verlag
- Fields, B., Paternò, F., Santoro, C (1999). Analysing User Deviations in Interactive Safety-Critical Applications, Proc. of DSV-IS '99, pp. 189-204, Springer-Verlag.
- Gamma, E., Helm, R., Johnson, R. Vlissides, J. (1995). Design Patterns: Elements of Object-oriented Software. Book published by Addison-Wesley
- Hartson, R., and Gray, P. (1992). Temporal Aspects of Tasks in the User Action Through Product and Process. New York. John Wiley.
- HAZOP (1996). MOD: Studies on systems containing programmable electronics. UK Ministry of Defence Interim Def Stan 00-58, Issue 1. Available from http://www.dstan.mod.uk/dstan_data/ix-00.htm
- Hix, D. and Hartson, H. R. (1993) Developing User Interfaces.
- Hollnagel, E., (1991). The Phenotype of Erroneous Actions: Implications for HCI Design. In: Weir, G.R.S. and Alty, J.L., (Eds.), Human-Computer Interaction and Complex Systems, Academic Press.
- IBM (1989) Common User Access: Advanced Interface Design Guide. IBM, SC26-4582-0
- Johnson, P., Johnson, H. (1989). Knowledge Analysis of Task: Task Analysis and Specification for Human-Computer Systems. In: Downton, A. (ed.): Engineering the Human Computer Interface. McGraw-Hill, Maidenhead 119-144
- Jordan, B. (1996). Ethnographic Workplace Studies and CSCW. In D. Shapiro, M.J. Tauber and R. Traunmueller (eds). The design of computer supported cooperative work and groupware systems. North-Holland, Amsterdam, 17-42.
- Kirwan, B (1994). A guide to practical human reliability assessment. Taylor and Francis.
- Nardi, B. (1995). Context and Consciousness: Activity Theory and Human Computer Interaction. MIT Press, Cambridge MS.
- Navarre D., Palanque P., Bastide R. Paternò F. & Santoro C. (2001). A tool suite for integrating task and system models through scenarios. In 8th Eurographics workshop DSV-IS'2001 LNCS, no. 2220. Springer, 2001

- Norman Donald A. (1988). *The design of everyday things* New York: Currency-Doubleday, 1988.
- Palanque P, Bastide R. & Paternò F. (1997). Formal Specification as a Tool for Objective Assessment of Safety-Critical Interactive Systems Interact'97 conference, Chapman et Hall. pp. 463-476.
- Palanque Ph. & Bastide R. Synergistic modelling of tasks, system and users using formal specification techniques. *Interacting With Computers*, Academic Press, 9,12, pp. 129-153
- Paternò, F. (1999) *Model Based Design and Evaluation of Interactive Applications*. Springer Verlag, Berlin
- Paternò F. and Santoro C. (2002). Preventing user errors by systematic analysis of deviations from the system task model. *International Journal Human-Computer Studies*, Elsevier Science, Vol.56, N.2, pp. pp. 225-245, 2002.
- Pocock, S., Fields, B., Harrison, M and Wright, P. (2001). *THEA – A Reference Guide*. University of York Computer Science Technical Report 336, 2001.
- Rasmussen, J. (1983). Skills, rules, knowledge: Signals, signs, and symbols and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):257-267
- Reason, J. (1990). *Human Error*, Cambridge University Press
- Smith, D.J., (2002). *Human Error (and how to prevent it)* e-learning resource. Available from <http://www.smithsrisca.demon.co.uk/unitHE2.html>
- Scapin, D., Pierret-Golbreich, C. (1989) Towards a Method for Task Description: MAD. In: Berlinguet, L., Berthelette, D. (eds.): *Proc. of Conf. Work with Display Units WWU'89*, Elsevier Science Publishers, Amsterdam (189) 27–34
- Sinnig D., Forbrig P. and Seffah A. (2003). Patterns in Model-Based Development, Position Paper in INTERACT 03 Workshop entitled: Software and Usability Cross-Pollination: The Role of Usability Patterns.
- Sinnig, D., Gaffar, A., Seffah, A., Forbrig, P. (2004). Patterns, Tools and Models for Interaction Design. MBUI Workshop 2004, P.09
- Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communications*. Cambridge, UK: Cambridge University Press.
- van der Veer, G., van der Lenting, B.F., Bergevoet, B.A.J. (1996) GTA: Groupware Task Analysis - Modeling Complexity. *Acta Psychologica* 91 297–322.
- van Welie M., van der Veer G.C., Eliëns A. (2000). Patterns as Tools for User Interface Design: In: *International Workshop on Tools for Working with Guidelines*, pp. 313-324, 7-8 October 2000, Biarritz, France.