



HAL
open science

Designing and Assessing Interactive Systems Using Task Models

Célia Martinie, Philippe Palanque, Marco Winckler

► **To cite this version:**

Célia Martinie, Philippe Palanque, Marco Winckler. Designing and Assessing Interactive Systems Using Task Models. Kronbauer, Arthur; Mattos, Ecivaldo; Sampaio, Andreia Libório; Boscaroli, Clodis. Book of Tutorials of the 14th Brazilian Symposium on Human Factors in Computing Systems (IHC 2015), Salvador Brazil, November 3-6, 2015, Brazilian Computing Society, pp.29–58, 2015. hal-03209324

HAL Id: hal-03209324

<https://hal.science/hal-03209324>

Submitted on 4 Apr 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Chapter

2

Designing and Assessing Interactive Systems Using Task Models

Célia Martinie, Philippe Palanque, Marco Winckler

Abstract

Task analysis is meant to identify user goals and tasks when using an interactive system. In the case of users performing real-life work, task analysis can be a cumbersome process gathering a huge amount of unorganized information. Task Models provide a mean for the analyst to organize information gathered during task analysis in an abstract way that can be further detailed if needed. This chapter aims at providing newcomers with background in task modeling. It provides an overview on how the recent advances in task description techniques can be exploited to design and assess interactive systems. As task models can be large it is important to provide the analyst with computer-based tools for editing task models, for analyzing them and for simulating them. For that, the HAMSTERS notation and its eponym modeling tool are used in this chapter to illustrate the examples issued from real industrial projects.

Resumo

A análise de tarefas visa identificar objetivos e tarefas de usuários durante a utilização de sistemas interativos. Em situações reais de trabalho, a análise de tarefas pode ser uma atividade densa devido a quantidade de informações obtidas de maneira desorganizada durante o processo. Modelos de Tarefas oferecem ao analista uma maneira de organizar as informações obtidas durante a análise forçando uma representação abstrata que pode ser detalhada e analisada na medida do necessário. O objetivo deste capítulo é fornecer um referencial para iniciantes sobre modelagem de tarefas. Ele apresenta como os recentes avanços em técnicas de descrição de tarefas podem ser empregados para projetar e avaliar sistemas interativos. Visto o tamanho avantajado de modelos, é importante que o analista possa dispor de ferramentas de edição, de análise e de simulação de modelos de tarefas. Por esta razão, a notação HAMSTERS e a ferramenta de modelagem homônima são usados aqui para ilustrar exemplos de modelo de tarefas obtidos à partir de estudos reais em projetos industriais.

2.1. Introduction

Task analysis is meant to identify user goals and activities when using an interactive system. In the case of users performing real-life work, task analysis can be a cumbersome process gathering a huge amount of unorganized information store in different formats such as paper documentation, text and video (from interviews), transcripts from scenarios,

Task Models (TM) consist in a representation of such information and provide a mean for the analyst to store information in an abstract way that can be further detailed and analyzed if needed. A task model can be of various forms starting from informal textual descriptions until formal models. Such models allows User Centered Designers to record in a systematic, complete and unambiguous way the set of user goals and the way those user goals can be performed on an interactive system. Reasoning about the Task Models supports the assessment of effectiveness of an interactive system (which is one of the most difficult dimension of usability to assess).

Task models have also proven being of great help for structuring user documentation, designing and assessing a training program, assessing the complexity of the users' work. If used for analysis, they can also provide support for identifying types, location and likelihood of human errors. When used for design they also provide precious support for identification of good candidates for task migration towards automation.

Several notations and methods have been introduced during the last fifty years. Each of them has been developed to provide support for analyzing user tasks in a particular context or during a particular stage of the design, development, or usage of an interactive system. For example, the HTA (Hierarchical Task Analysis) task analysis method and notation was coined in the late 1960's [Meyer et al. 1967, Annett 2004] in order to provide support for understanding the skills required in complex non repetitive operator tasks (for the steel production industry). Since this seminal work, other methods and notations have been developed for various purposes such as providing support for task centered system design [Greenberg 2004] [Mori et al. 2002], estimating human performance [Kieras 2004], automatic generation of interactive applications [Paterno et al. 1998] and taking into account potential human errors at design time [Paterno and Santoro 2002].

The HAMSTERS notation and its eponym tool has initially been designed to provide support for ensuring consistency, coherence and conformity between user tasks and interactive systems at the model level [Barboni et al. 2010]. It has then been further enhanced and now encompasses notation elements such as a wide range of specialized tasks types, data and knowledge explicit representations, device descriptions, genotypes and phenotypes of errors, collaborative tasks among others. The main elements of this notation and associated tool, as well as an example on how to use them, are presented in this chapter. Beyond that, this chapters also contains:

- The benefits of using task modeling techniques to design, structure and assess UIs,
- A method and its associated process to describe users' activities in a systematic and structured way,
- An detailed example describing how to build a task model and how to analyze it,
- How to use the HAMSTERS tool suite for editing, analyzing and simulating task models.

This chapter is composed of five main sections, in addition to the introduction (section 2.1) and conclusion (section 2.7). Section 2.2 introduces the purpose of task modeling and task models. It highlights the differences with close concepts such as task analysis, user models and system models. Section 2.3 presents the main elements of HAMSTERS notation. Section 2.4 presents the HAMSTERS CASE (Computer-Aided Software Environment) tool. Section 2.5 shows how the notation and tool can be used with the illustrative example of the usage of an ATM (Automated Teller Machine) application. At last, section 2.6 gives an insight on advanced usage of task models for designing and developing interactive systems.

2.2. What should (not) be represented in a task model?

As explained above information to be added in the task models is the result of the task analysis activity. Task analysis and task model must remain two distinct activities even though they might inform and influence each other as presented in section 2.2.1.

When describing users' activities by means of a task model it is usually cumbersome to only integrate information related to these users activities. Indeed, as the task will be carried out by human it is easy to integrate internal information processing (for instance) thus bending the task model towards a human model. Such aspect and how to avoid that pitfall is presented in section 2.2.2. The same holds the computing system aspects. Indeed, tasks might be performed using a computing system and one can be tempted to integrate state representation, detailed behavior, events ... that should remain in a model of the computing device and not in the task models. This aspect is addressed in section 2.2.3 which aims at defining which aspects of the computing device has to be represented in the task model and what must remain outside. As scenarios are widely used in the field of User Centered Design (see for instance Scenario-based design approach proposed in [Rosson & Carroll 2001]), section 2.2.4 defines the difference between a task model and a scenarios as well as how these two artefacts relate to each other.

2.2.1 Task models versus task analysis

As stated by Annett (Annett et al. 2004) "Analysis is not just a matter of listing the actions or the physical and cognitive processes involved in carrying out a task, although it is likely to refer to either or both. **Analysis**, as opposed to **description**, is a procedure aimed at identifying performance problems and proposing solutions." This clearly defines the border between so called description activities and analysis ones. In that sentence however there is a confusion between analysis of the work of users ("standard" task analysis) and the analysis of the descriptions/models. These descriptions/models are the result from the activity of organizing information gathered while performing task analysis. Such description/models can, in turn, be analyzed in order to identify missing, redundant or inconsistent information or in order to identify better organization of work i.e. redefine users goals and tasks, allocate tasks differently between users, ...

Another confusion can be found in [Diaper 2004] when stating "Task analysis produces one or more models of the world and these models describe the world and how work is performed in it".

Of course task analysis and task modeling are activities that should be intertwined and task models grow when task analysis progress.

2.2.2 Task model versus user model

As task models describes how users reach their goals and perform their activities, information such as “user reads information on a display” or “user analyses the value presented on a display” is included in the task model. Starting from this one might be interested in representing (in the task model) the fact that the user will store this information in the short term memory, make some efforts to remember it and after a while will forget it. Such information should not be included in the task model as it belongs to human information processing activities and, as such, should appear in a user model describing perceptive, motor and cognitive activities at a generic level (for every human) as well as at a specific level (for the target user group).

If such a user model is built as part of the modeling process (as for instance this is the case of computer supported learning environments) the task of the analyst will be to check that the information represented in these two models are consistent and contain as little redundancy as possible. Failing to so will end up with user and task models being highly coupled which require replication of modifications in both models thus increasing significantly analysts’ workload.

2.2.3 Task model versus computing system model

Modelling activities is one of the corner stone of computer science as this is the only way to handle the complexity of these systems. Many modeling techniques and notations have been proposed in that domain reaching a climax with the proposal of UML [OMG 2003] and more recently SysML where 11 different notations are introduced for describing computer systems. Among all of these notations one targets at describing user activity and is called “use cases”. Those “use cases” are very different from task models even though, at a high level of abstraction, they aim at the same objective. Describing commonalities and differences between task models and use cases is beyond the scope of this tutorial but the interested reader can find detailed information in [Sinnig et al. 2013].

One of the pitfalls of task modelling is to avoid representation in the task models of information that should appear in one of the computer system models. This is not an easy job as the task model needs to represent information about interaction i.e. the activities of the user that are performed triggering commands in the underlying system. As far as the interactive aspects of the computing system are concerned information such as events, states, graphical rendering, ... should remain in the system model. However, user interface components triggering (e.g. UI button triggering or entering a value) should explicitly appear in the task models as they are parts of the user activity. Information processing by the computing system might also need to be represented if they result in providing feedback to the user or if they need time imposing waiting time on the user side.

This distinction between system model and task model will be explicitly addressed in the case study section.

2.2.4 Task model versus scenarios

As defined by [Rosson & Carroll 2001], scenarios “consist of a setting, or situation state, one or more actors with personal motivations, knowledge, and capabilities, and various tools and objects that the actors encounter and manipulate. The scenario describes a sequence of actions and events that lead to an outcome. These actions and events are

related in a usage context that includes the goals, plans, and reactions of the people taking part in the episode”. Task models consist in an abstract description of user activities structured in terms of goals, sub-goals and activities. [Anderson et al. 1990]. The table below summarizes side-by-side differences between scenarios and task models:

Scenario	Task model
Concrete	Abstract
Flat (like a storyline)	Hierarchical (from more abstract to more concrete)
Incomplete (only represent one execution amongst many)	Exhaustive (represent all the tasks of interest)
Instances (scenarios contains the values)	Variables (only variable names are represented (possibly values on conditions))
Linear (only one story is described)	Branching (all the alternatives of activities are represented)
Explicit (all the relevant information is given)	Implicit (all details are abstracted away)
Quantitative (time, number of resources ...)	Qualitative (ordering of activity, type of information needed, ...)
Practical (time, number of resources, ...)	Theoretical (errors are not represented)
Borderline (represent cases at the limit)	Mainline (represent the standard, usual activity)

CTTE integrates mechanisms from support task model construction by exploiting textual informal scenarios [Paterno & Mancini 1999]. The text is analyzed in order to identify verbs (that will correspond to tasks), actors (and for each of them a task model will be built) and nouns (that will correspond to objects in the task model). Such systematic exploitation of data-related information has been for the first made explicit in [Caffiau et al. 2010].

Scenarios can be produced by the execution of task models as initially proposed in CTTE [Mori et al 2002]. Each time the task model is simulated (for instance in order to test that the model corresponds to the activity of the operators) the execution is recorded as a scenario. Execution of interactive systems by means of scenarios produced from a task model has been proposed in [Navarre et al. 2001] where scenarios (such as the ones described right before) are used as input for the autonomous execution of the system (no user input is required as this information is already in the scenario). This makes it possible to assess consistency between the information described in the task model and the actual behavior of the interactive system.

2.2.5 Potential uses of task models and expected benefits

Task models are considered by many as a corner stone in the development process of usable interactive applications. Indeed, they provide a unique mean for gathering information about users’ roles, goals and activities either being about an extant or envisioned system. However, at the same time, task models are also considered as cumbersome, expensive to build and mainly useful in the early phases of the development process. When used throughout the development process as well as at operation time (when the system has been deployed and is currently used) task models bring many benefits such as:

- Support the assessment of the effectiveness factor of usability by identifying which tasks are supported by the interactive application and which ones are not;
- Support the assessment of the task complexity in terms of perception, analysis, decision and motor action of users in order to reach a goal [Fayollas et al. 2014],

assessment of operators' performance to reach a goal [Sweargin et al. 2013] which can lead to predictive workload assessment [O'Donnell & Eggermeir 1986];

- Support the construction of training material and training sessions of operators of complex systems [Martinie et al. 2011b];
- Support the structuring and the construction of user documentation [Gong & Elkerton 1990];
- Support the heuristic evaluation of usability of interactive applications (better than when task models are not used) not only for single user applications [Cockton & Woolrych 2001] but also for multi user applications [Pinelle et al. 2003],
- Support the identification of user errors and their impact on the overall performance for reaching the goals [Palanque & Basnyat 2004] as well as preventing those user errors [Paterno & Santoro 2001];
- Support the identification of tasks that are good candidate for migration towards an automation of the system [Martinie et al. 2001c] but also towards other users in the context of collaboration [van Welie & van der Veer 2003];
- Makes it possible to provide users with contextual help i.e. explicit information about how (which tasks to perform) to reach the goal both at design time [Pangoli & Paterno 1995] and from the current state of interaction while interacting with the system [Palanque & Martinie 2011];
- Support the redesign of the extant system by analysis of extant task models and producing task models for the future system as promoted in ADEPT framework [Wilson et al. 1996].

It is important to note that most of the potential benefits listed above require that the interactive application and the information represented in the task model are compatible. Such compatibility can be seen at lexical (for each interface object corresponds a low level task in the task model and reciprocally), syntactic (the task structure and temporal operators are conformant with the availability of interface objects i.e. compatible with the dialogue of the UI) and semantic levels (the interactive application allows users to reach the goals identified in the task model) [Palanque et al. 1995].

The tutorial argues that it is possible to **ensure compatibility between a task model and an interactive application**. Beyond previous work in which compatibility was ensured by generation of the application from the task model (as promoted in [Wilson et al. 1996]) or by “connecting” a model of the application with the task model (as promoted in [Palanque et al. 1995] and [Barboni et al. 2010]). More recent work [Martinie et al. 2015] has also demonstrated that HASMTERS and its framework allow coupling task models with any existing already in use, interactive application.

2.3 Foundations of HAMSTERS notation

HAMSTERS (Human – centered Assessment and Modeling to Support Task Engineering for Resilient Systems) is a tool-supported graphical task modeling notation for representing human activities in a hierarchical and structured way. A HAMSTERS task model is a graphical tree of nodes that can be tasks or temporal operators.

2.3.1 Hierarchical structuring

At the higher abstraction level, the main goal of the user is represented as the top node of a top-down tree of nodes. This goal is refined into sub-goals which can in turn be decomposed into activities (as illustrated in Figure 2. 1).

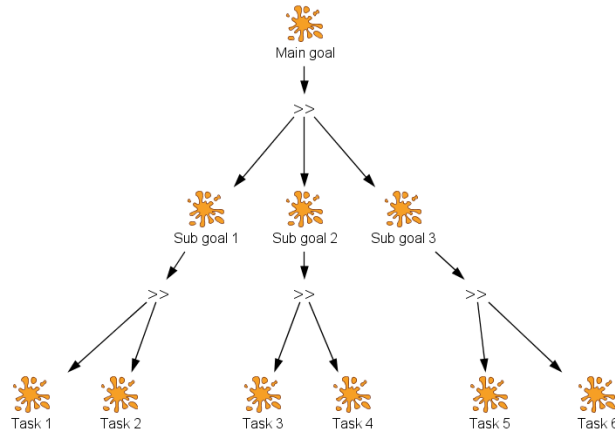


Figure 2. 1. Goal hierarchy in a task model

A goal can be refined into sub-goals by answering the question “How?”. A sub-goal can also be refined into tasks by answering the question “How?”. In the other way round, tasks can be gathered as belonging to a sub-goal by answering to the question “Why?”. These read direction for a hierarchical task model are illustrated in Figure 2. 2.

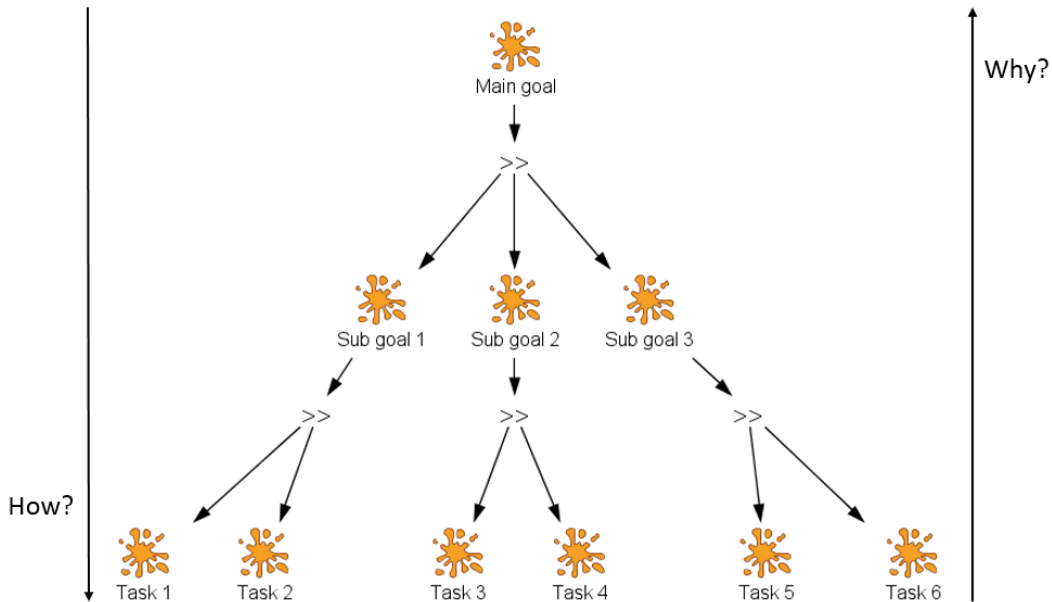


Figure 2. 2. Read directions in a hierarchical task model

2.3.2 Temporal ordering of tasks

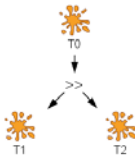
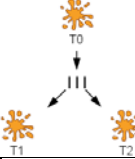
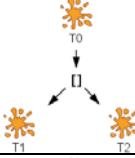
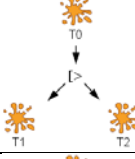
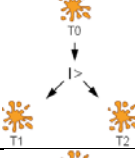
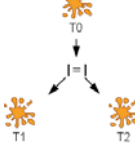
Temporal operators (depicted in Table 2. 1) are used to represent temporal relationships between sub-goals and between activities. We use the temporal operator from the LOTOS notation as introduced in [Wilson et al. 1993] which have been largely adopted thanks to CTT notation and CTTE environment [Paterno 1999].

Table 2. 1 Operator types within HAMSTERS

Operator type	Symbol
Enable	>>
Concurrent	
Choice	
Disable	>
Suspend-resume	>
Order Independent	=

Table 2. 2 presents how temporal operators are used to explicit the temporal ordering between tasks.

Table 2. 2 Illustration of the operator types within HAMSTERS

Temporal operator in a task model	Operator type	Description
	Enable	In order to accomplish T0, T2 is executed after T1.
	Concurrent	In order to accomplish T0, T1 and T2 are executed at the same time.
	Choice	In order to accomplish T0, T1 is executed OR T2 is executed
	Disable	In order to accomplish T0, execution of T2 interrupts the execution of T1
	Suspend - resume	In order to accomplish T0, execution of T2 interrupts the execution of T1, T1 execution is resumed after T2.
	Order independent	In order to accomplish T0, T1 is executed then T2 OR T2 is executed then T1
















2.3.3 Task types

Tasks can belong to a type amongst four:

- Abstract: an abstract task (represented by the orange icon in row 1 in Table 2. 3) may be a goal or a sub-goal that can be refined or a task that in the current state of analysis is not known as being performed by the user or by the system.

- User: a generic task describing a user activity. A User abstract task (row 2 in Table 2. 3) can be put in a task model to specify that this user task has to be refined. A user abstract task can be specialized (from left to right in row 2 in Table 2. 3) as Perceptive task (e.g. reading some information), Motor task (e.g. pressing a button), User task (encompassing perceptive, motor and cognitive activities, e.g. taking the train) or Cognitive task (e.g. comparing value, remembering information).
- Interactive: a task describing an interaction between the user and the system (row 3 in Table 2. 3). An abstract interactive task (row 3 in Table 2. 3) can be put in a task model to specify that this interactive task has to be refined. It can be refined (from left to right in row 3 in Table 2. 3) into Input task when the users provide input to the system, Output task when the system provides an output to the user and Input/Output task (both but in an atomic way).
- System: a task describing a function that is performed by the system. An abstract system task (row 4 in Table 2. 3) can be put in a task model to specify that this system task has to be refined. It can be refined (from left to right in row 4 in Table 2. 3) into Output task when the system provides an output to the user, Input task when the users provide input to the system and Input/Output task (both but in an atomic way).

Table 2. 3 Task types in HAMSTERS

	Abstract	Input	Output	I/O	Processing
Abstract	 Abstract	Not Applicable	Not Applicable	Not Applicable	Not Applicable
User	 User abstract	 Perceptive	 Motor	 User	 Cognitive
Interactive	 Abstract interactive	 Input	 Output	 Input/Output	Not Applicable
System	 Abstract system	 Output	 Input	 Input/Output	 System

Cognitive user tasks have been refined according to the Parasuraman model of human information processing [Parasuraman et al. 2000] in two cognitive subtypes: analysis and decision.

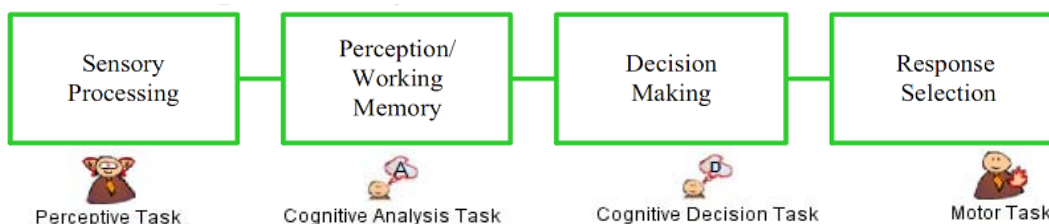


Figure 2. 3. Refinement of user tasks (according to [Parasuraman et al. 2000])

2.3.4 Task properties

Tasks may be optional, iterative or both optional and iterative. The representation of these properties is depicted in Figure 2. 4.



Figure 2. 4 Representation of properties for optional and/or iterative tasks

HAMSTERS provide support to associate minimum and maximum execution time to a task (as shown in Figure 2. 5). In this way, it enables:

- Checking temporal relevance between user’s activities and system information processing.
- Validating the developed system w.r.t. users’ performances evaluation with usage scenarios.

Grab card - Properties	
Properties	
description	Grab card
details	
criticality level	0
optional	<input type="checkbox"/>
iterative	<input type="checkbox"/>
number of iteration	0
type	USER
position	0
minimum execution time	500
maximum execution time	5000

Figure 2. 5 Quantitative temporal properties of a task (excerpt from the task properties editor)

2.3.5 Information representation in task models

HAMSTERS expressive power goes beyond most other task modeling notations particularly by providing detailed means for describing data that is required and manipulated [Martinié, Palanque, Ragosta, Fahssi 2013] in order to accomplish tasks. Information (“Inf:” followed by a text box) may be required for execution of a system task, but it also may be required by the user to accomplish a task. Physical objects required for performing a task can also be represented (“Phy O”) as well as the device (input and/or output) with which the task is performed (“i/o D”).

Inf :	Information (user side)	i/o D :	Input device
Phy O :	Physical object (user side)	i/o D :	Output device
Phy O :	Physical object (system side)	i/o D :	Input/Output device
Obj :	Object (system side)	Sw A :	Software Application

Figure 2. 6 Representation of Objects, Information, I/O devices and software applications

Figure 2. 7 illustrates the relationships (input, output) between data and tasks that can be expressed with HAMSTERS notation. Objects (data, information...) can be needed as an input to accomplish a particular task (incoming arrow from data to task). Particular tasks may generate an object or modify it (outgoing arrow from task to data). Required Input and/or output devices are represented and connected to an interactive task thanks to a thin line.

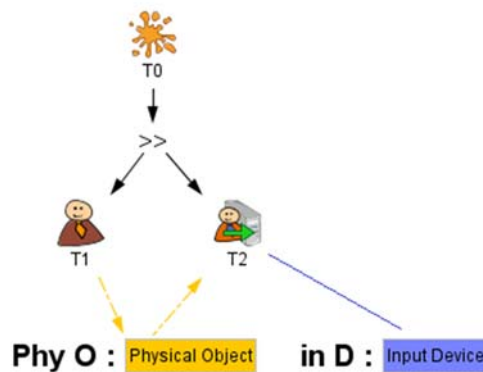


Figure 2. 7 Relationships between data and tasks

2.3.4 Human knowledge

Two main types of knowledge that have to be taken into account when analysing users' activities. The first one is declarative knowledge that can be structured and grouped using concepts (objects or information from the outside world and knowledge from the user). The second main type of knowledge is procedural knowledge, which can be represented in a hierarchical structure. Two sub-types of knowledge should also be taken into account when analysing users' activities: situational and strategic. Both sub-types help refining declarative and procedural knowledge to represent more precisely users' activities. The integrative approaches suggest that declarative and procedural knowledge can be structured, grouped and connected together as they are closely intertwined while users perform tasks

Knowledge required to accomplish a task is explicitly represented in a HAMSTERS task model. Knowledge of declarative type and its refinement in strategic and situational dimensions can be represented using the corresponding boxes illustrated in Figure 2. 8. Relationships between the represented knowledge and the tasks can be represented using input/output relationships represented with arcs as for the objects.

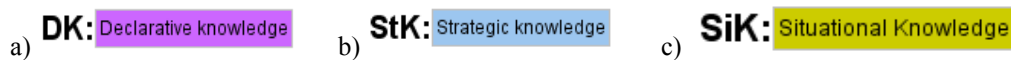


Figure 2. 8 Representation of a) declarative knowledge which can be further refined into b) strategic declarative knowledge and c) situational declarative knowledge

Representative distinctions between strategic and situational procedure can be made using two new types of arcs illustrated in Figure 2. 9. An ordered set of actions related to a strategy the user can apply will be highlighted with the blue "St" tagged arcs (Figure 2. 9a)). An ordered set of actions the user can execute in a given situation will be highlighted with the green "Si" tagged arcs (Figure 2. 9b)).

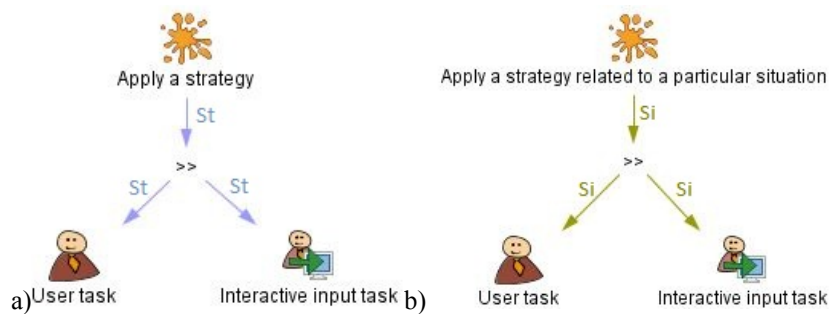


Figure 2. 9 Representation of procedural knowledge refined into a) strategic procedural knowledge and b) situational procedural knowledge

2.3.5 Structuring mechanisms

Complexity in models is a recurrent problem with model-based approaches that might require significant availability of resources which is sometimes perceived as too much effort, too long to produce and not being cost effective enough. This might be true if every model has to be developed from scratch each time a new application is considered and if the modeling techniques are not equipped with adequate tool support. Moreover, beyond tool-support, model complexity is also a concern at the notation level.

Abstraction and refinement of task models is not sufficient for handling large real world applications [Martinie, Palanque and Winckler 2011]. For that three mechanisms (sub-models, sub-routines and components) enable to deal with complexity in task models. These three mechanisms aim at supporting rapid task-model development by structuring models and improving reuse of existing models.

Sub-model

Sub-models are based on the refinement/abstraction principle and make possible to define elementary reusable bricks in task models. A large task model can thus be decomposed into several duplications of elementary tasks (called sub-models). These sub-models can then be reused (as a kind of “copy”) in various places of the same model and even in other models. Each time one of the attributes of these elementary sub-models is modified, the modification is reflected in all the other “copies” of the same sub-model.

While task notations propose reuse at the class level (an example of such a class being a “motor task type”) the sub-model proposes reuse at the instance level. For example, if in a task model, a task “push button” appears many times (because moving the lever can be performed by users for reaching multiple goals, such as to change gears in a car to reduce or to increase velocity) the sub-model construct makes it possible to handle those instances altogether.

Figure 2. 10 illustrates sub-models in HAMSTERS. Copy tasks (such as the bottom left one “T1.2”) is the same sub model as the task with the same name under the “T2” abstract task. All the properties of these two leaf tasks are shared and changing the type or name of one is immediately reflected on the other one.

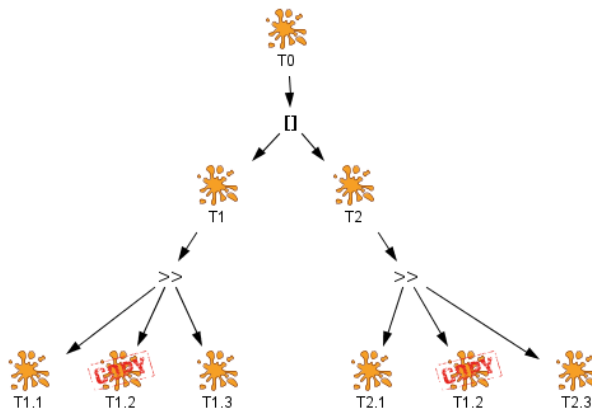


Figure 2. 10 Representation of copy tasks

Subroutine

Subroutines are used to structure task models and to define information passing between task models. This mechanism is similar to procedure calls in programming languages and parameterization of the behavior is possible via input and output parameters.

The subroutines aim at reusing a sub-tree in a task model. A group of tasks (represented as a tree) might have to be performed in multiple occasions with very little differences which are depending on some values and represented as pre and post conditions. The subroutine makes it possible to describe recurring behaviors and to describe explicitly the parameters and how they influence the task model behavior.

A subroutine is a group of activities that users perform several times possibly in different contexts which might exhibit different types of information flows. Fig. 3 provides the description of sub-routines. The icons for input and output parameters are filled if values are needed and computed. In the HAMSTERS CASE tool, the sub-routines are stored as task models but are gathered in the project tree under the same grouping called “subroutines” (see Fig. 9 a)). The task models are stored under the grouping called “Roles” because they specify the tasks related to a certain role.

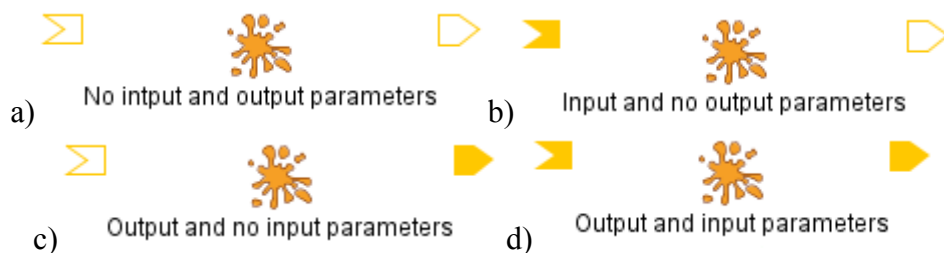


Figure 2. 11 Representations of subroutines in HAMSTERS

Figure 2. 1 shows an example of how a subroutine is described in a task model. Abstract task “T2.1” is a subroutine with an input parameter (Information “Information”) and no output parameter.

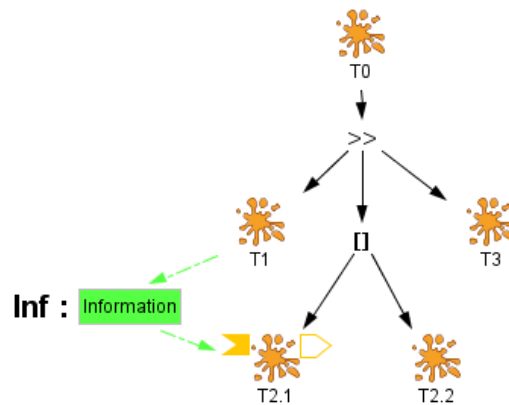


Figure 2. 12 Representation of a task model embedding a subroutine task

The refinement of task “T2.1” in Figure 2. 1 is described in another task model, which is depicted in Figure 2. 13.

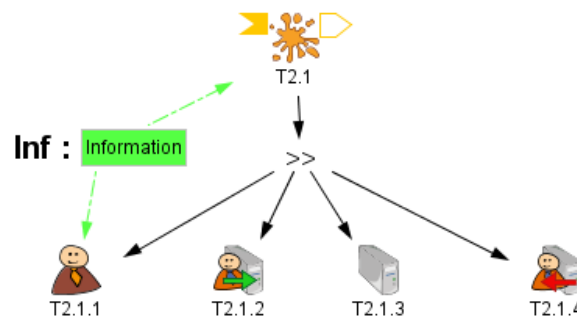


Figure 2. 13 Description of the refinement of T2.1 subroutine task (used in Figure 2.12)

Generic component

The goal of using generic components for task models is to allow for reuse of modeling efforts in a more general way than sub-routines. This structuring mechanism aims at defining and reusing generic features of sub-models. In addition to providing input and output parameters for tuning the reuse of task models at task model simulation time (in the same way as subroutines), generic components go beyond as they provide parameters that can be tuned at task model edition time. The representation of a generic component is depicted in Figure 2. 14.



Figure 2. 14 Representation of a component in HAMSTERS

Figure 2. 15 shows an example of how a subroutine is described in a task model. Goal T0 will be accomplished after the sequential execution of task T1 instantiated with (value of param1, value of param2) and of task T1 instantiated with (other value of param1, other value of param2).

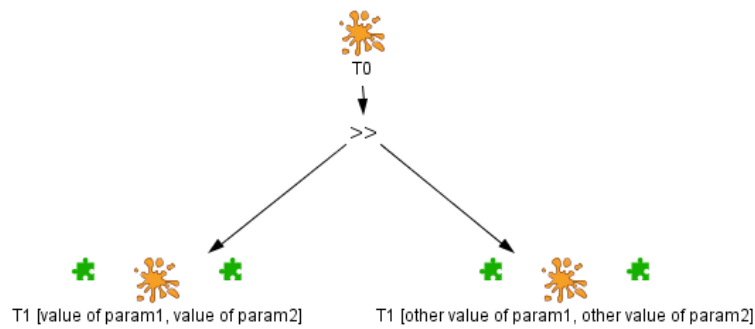


Figure 2.15 Representation of a task model embedding instances of a component

Figure 2.16 depicts the refinement of the generic component T1.

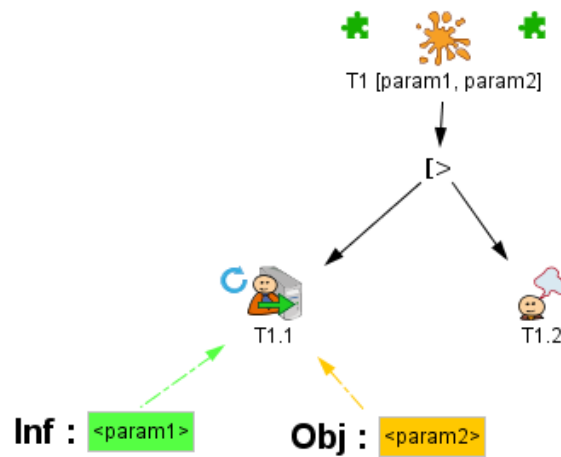


Figure 2.16 Description of the refinement of the T1 component

Figure 2.17 depicts the instantiation of the refinement of the generic component T1 with the values of the parameters (value of param1, value of param2).

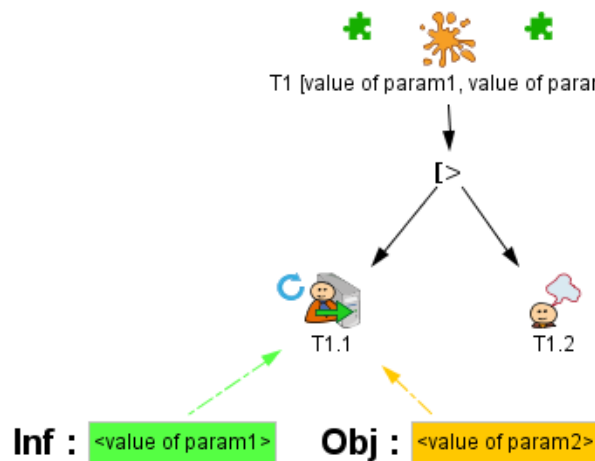


Figure 2.17 Description of the T1 component instantiated with a value for each parameter

2.3.6 Representation of collaborative tasks

Collaborative work is performed by several persons, each one having a role in the achievement of common goals. Collaborative work can be described at different abstraction levels: at the group level and at the individual level. A group task is a set of task that a group has to carry out in order to achieve a common goal [McGrath 1984], whereas a cooperative task is an individual task performed by a person in order to contribute to the achievement of the common goal [Roschelle and Teasley 1995].

In order to be able to describe group tasks, we introduce several new task types illustrated in Figure 2. 18.

These group tasks provide support for describing high level activities that a group of person have to accomplish:

- An abstract group task is a task that can be decomposed into user, system, interactive and collaborative tasks.
- A group (of users) task is task that can be decomposed in user and collaborative user tasks.
- An interactive group task can be decomposed in interactive and collaborative interactive tasks.
- A system group task can be decomposed in system tasks.

		Abstract	Input	Output	I/O	Processing	Group
Abstract			Not applicable	Not applicable	Not applicable	Not applicable	
User	Indiv.						
	Coop.						
Interactive	Indiv.					Not applicable	
	Coop.					Not applicable	
System							

Figure 2. 18 Task types including collaborative and group tasks

The refinement of group tasks into low-level activities needs fine -grain task types to describe individual and cooperative tasks that have to be performed in order to contribute to the group activities. As individual task types were already available within HAMSTERS, we then introduce cooperative tasks, illustrated in Figure 2. 18. A cooperative task is a task related to a role and accomplished in correlation with another cooperative task that relates to a different role. A cooperative task may be of various types within the user and interactive main family types.

Cooperative tasks may be performed within various space-time constraints (local/distant, synchronous/asynchronous) [Ellis et al. 1991]. These constraints can be described with notation elements illustrated in Figure 2. 19.





	Local	Distant
Synchronous	 Cooperative input task	 Cooperative input task
Asynchronous	 Cooperative input task	 Cooperative input task

Figure 2. 19 Elements of notation related to space-time constraints

Cooperative task may be dedicated to one or more of the following type of collaborative activities: production, coordination, communication. It is then possible to associate one or more properties amongst this set. For example, Figure 2. 20 a) shows that one task is dedicated to coordination whereas Figure 2. 20 b) shows that the task is dedicated to both coordination and communication.

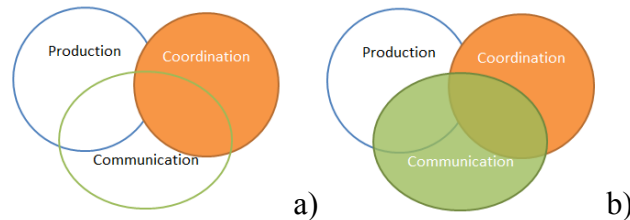


Figure 2. 20 Example of cooperative task properties from a “functional clover” [Calvary et al. 1997]

2.4 HAMSTERS in practice: the CASE tool and the task model building process

As task models can be large, it is important to provide the analyst with computer-based tools for editing task models and for analyzing them. To this end, the HAMSTERS task modeling tool provide support for creating, editing, and simulating the execution of task models.

2.4.1 Editing

As depicted in Figure 2. 21, HAMSTERS software tool for editing task models is composed of three main areas:

- On the left hand side, the project exploratory,
- In the center, the task model editing area,
- On the right hand side, the palette containing task types and temporal ordering operators.

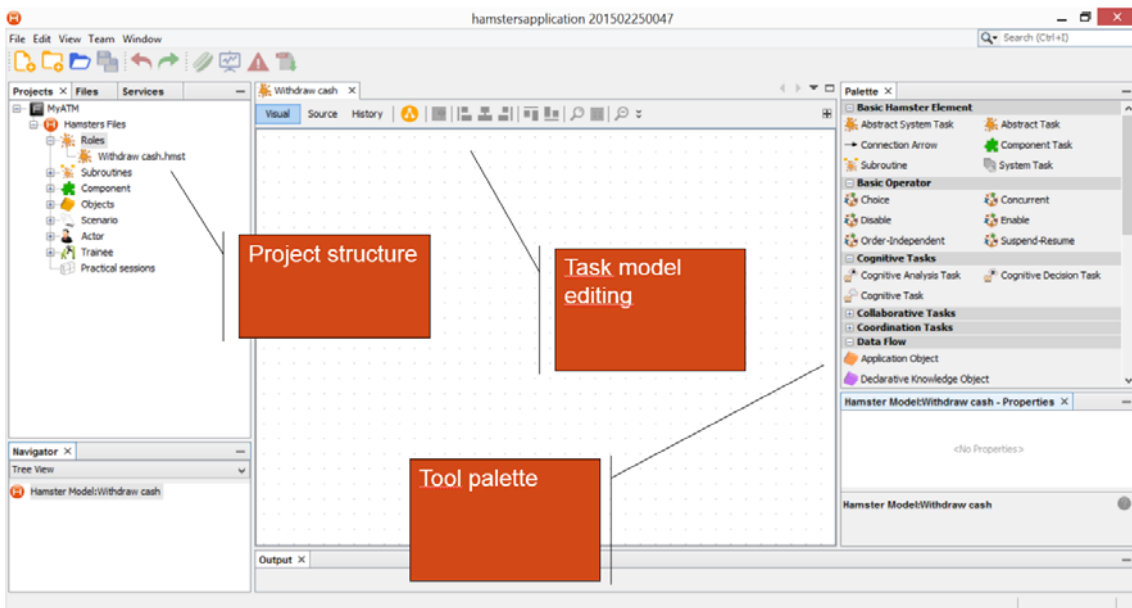


Figure 2. 21 HAMSTERS CASE tool

Visual notation elements can be added to the task model by a drag and drop operation from the palette to the task model area (Figure 2. 22).

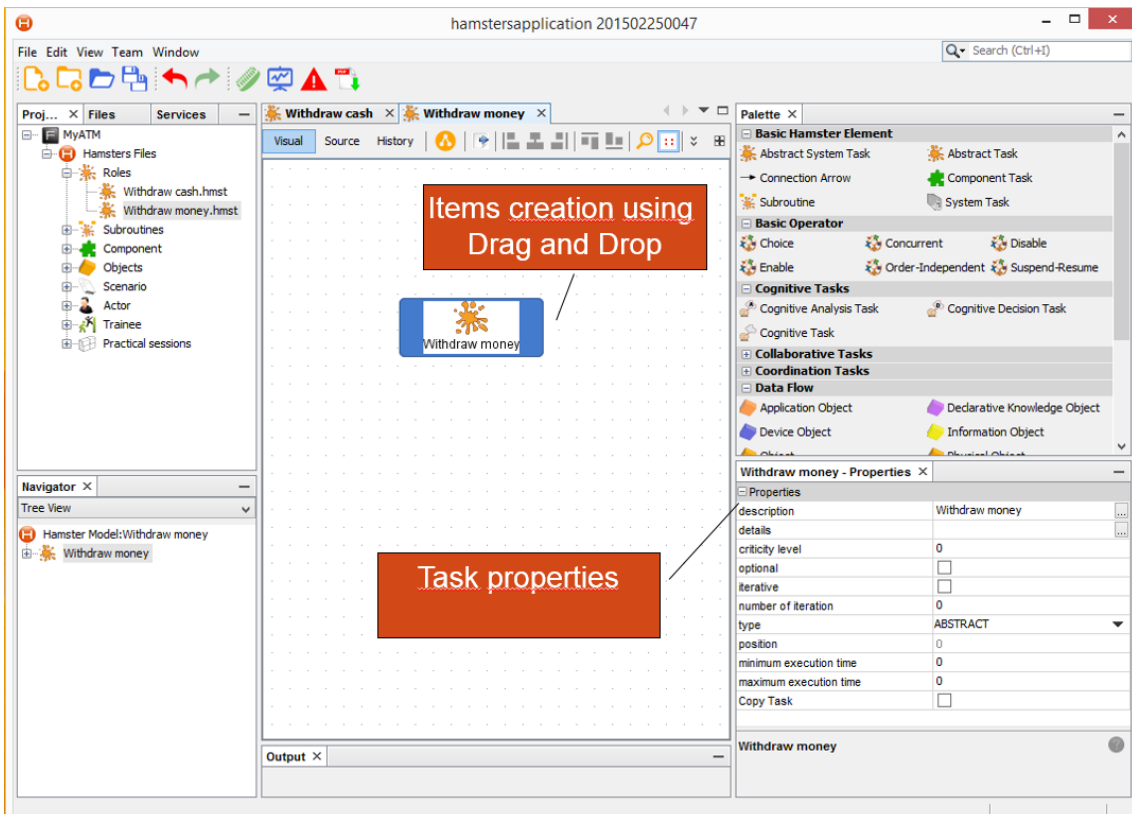


Figure 2. 22 Task model editing

The task properties panel provides support, on the bottom left in Figure 2. 22, provides support for viewing and editing task properties (such as estimated minimum and maximum time).

2.4.2 Simulation

The execution of task models can be launched from the project panel, by right-clicking on a task model and then by selecting the “Run simulator” menu option (depicted in Figure 2. 23). Once this menu option has been selected, a pop-up window appears to enable the user to choose a name for the scenario (depicted in Figure 2. 24). This scenario will contain the list of tasks that have been executed during the simulation.

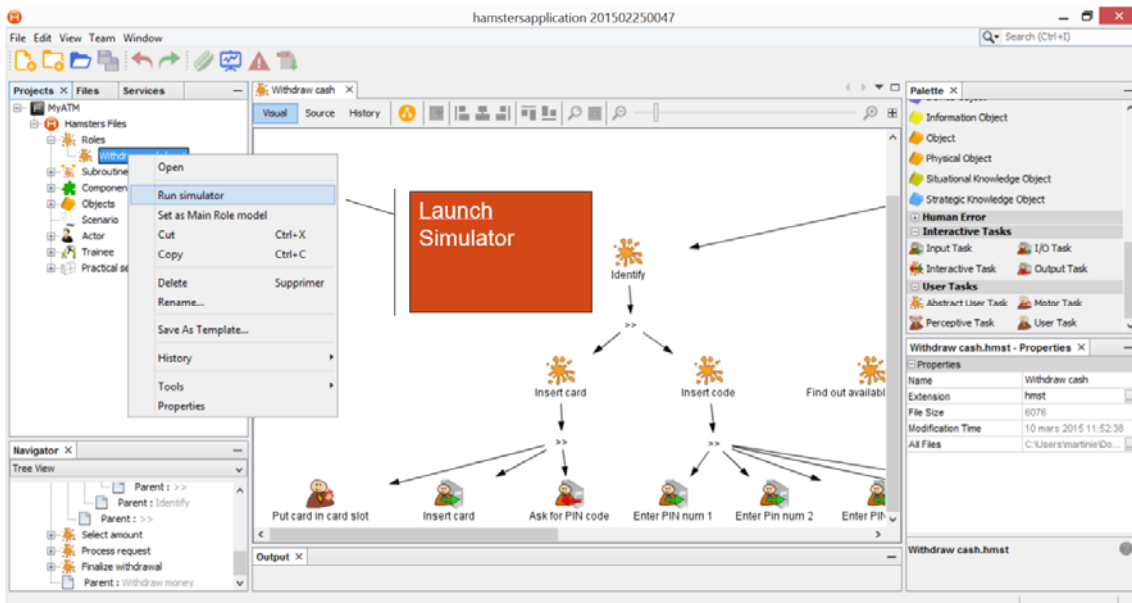


Figure 2. 23 Launching of task models simulation

Once a name has been chosen for the scenario, the simulation panel appears on the right hand side in the HAMSTERS software environment, as depicted in Figure 2. 25. At the same time, a new visual element appears in the project explorer panel, on the left hand side of the HAMSTERS software environment. This visual element represents the new scenario file that has been created (Figure 2. 25).

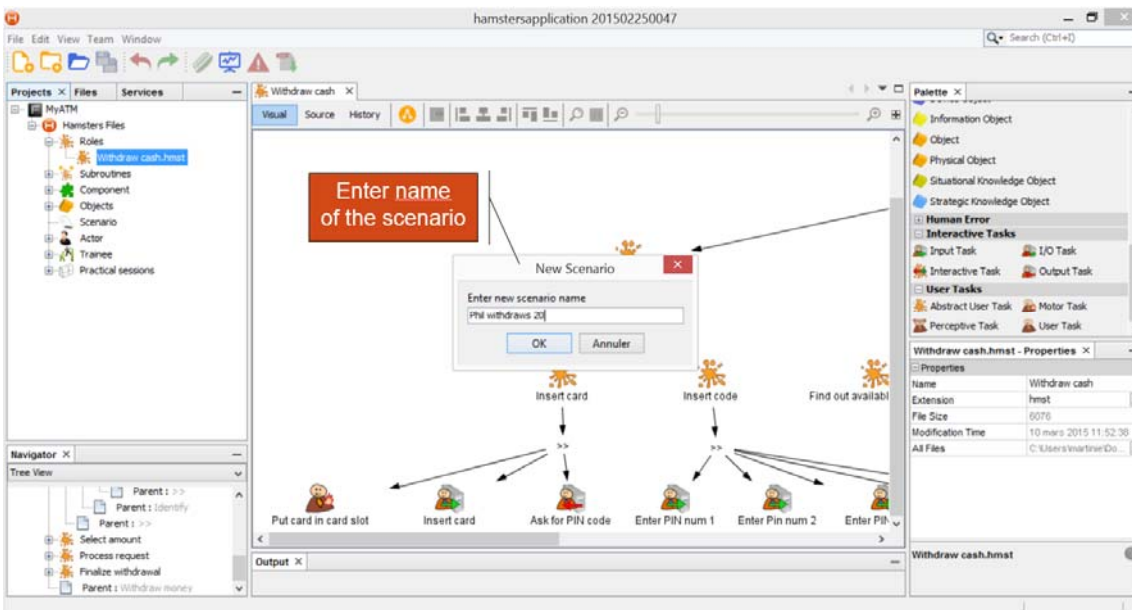


Figure 2. 24 Creation of a scenario

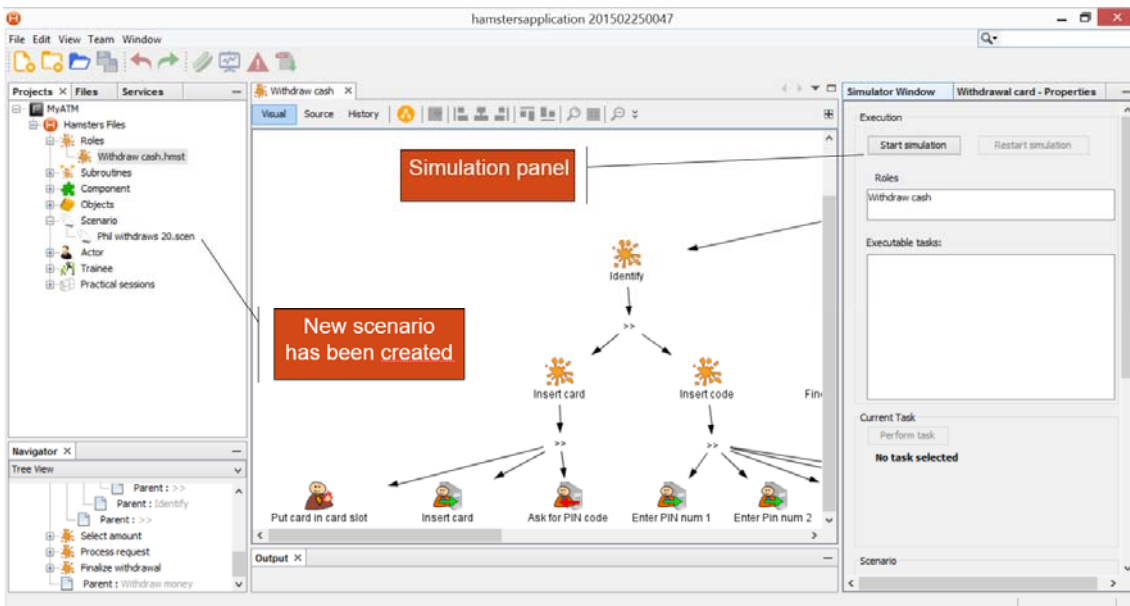


Figure 2. 25 Simulation panel

The simulation panel provides information about:

- the current tasks that are available for execution (list in the upper part of the simulation panel in Figure 2. 26)
- the scenario, i.e. the tasks that have been executed (list in the lower part of the simulation panel in Figure 2. 26)

The tasks which are available for execution are highlighted in green in the task model (in the central part in Figure 2. 26).

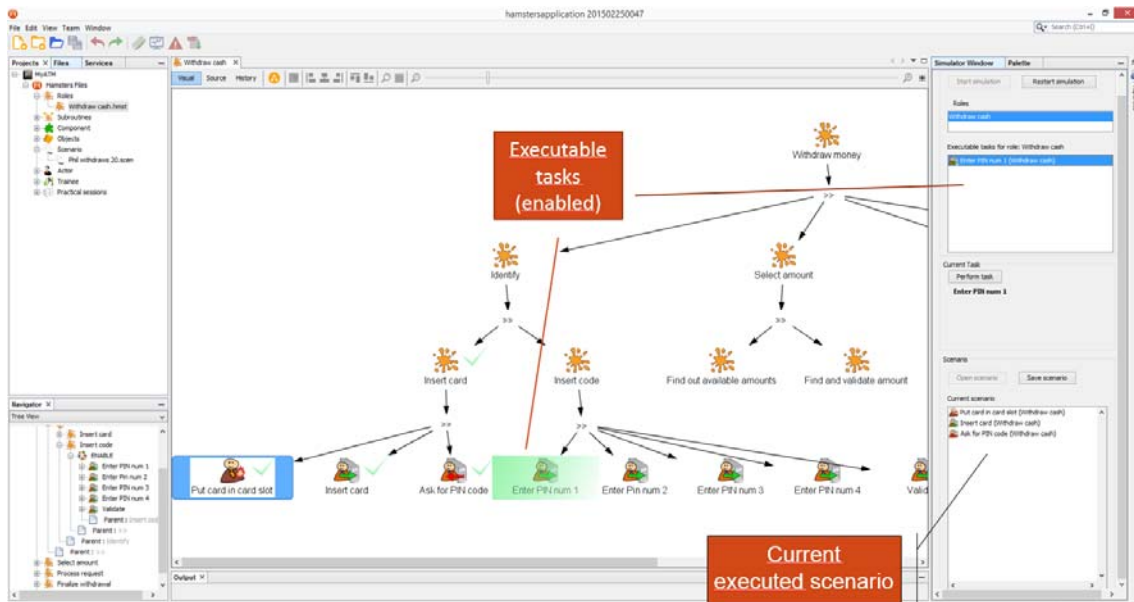


Figure 2. 26 Representation of executable and executed tasks during simulation

2.4.3 Building a task model

The aim of the task modelling activity will help in determining what to describe in the models. The selection of the tasks to be modeled and of the levels of refinement will be chosen according to the answers to the following questions: What is the purpose of the tasks models? What will they be used for?

Several methods have been proposed to analyze tasks and describe them. The following high level process is inspired from HTA [Anett 2004] and specialized according to the element of the HAMSTERS notation:

1. Gather information about main goals, sub-goals and their ordering
2. Format this gathered information in an initial version of the task models
3. Refine the task models by describing in details:
 - a. Actions that have to be accomplished
 - b. Data required to perform these actions
4. Implement appropriate structuration mechanisms
 - a. Use subroutines to avoid duplication of sets of tasks and increase legibility
 - b. Use generic component mechanisms to abstract set of actions that can be performed with a particular part of the user interface (independently from the related system function)

These sequence of modeling steps can be repeated until the task models are suitable for the purpose of the task analysis.

2.5 Illustrative case: ATM

The ATM (Automated Teller Machine) is an interactive system that most of the people use on a regular basis to withdraw money from their bank account. This illustrative example has been chosen because it is simple and meaningful. This system, usually available in public spaces, and the associated activities are known by the readers. The usage of this type of system requires the manipulation of several types of data. Figure 2. 27 shows a screenshot of the user interface of a Java version of an ATM.



Figure 2. 27 Screenshot of the ATM user interface

In this illustrative example, we aim at producing the task models of the activities required to be performed with the presented ATM interactive application in order to be able to withdraw money.

2.5.1 Initial task model

As described in Figure 2. 28, a sequence of tasks have to be performed with the ATM interactive application in order to withdraw money. First, the user has to identify, then s/he has to select an amount. Once the amount to be withdrawn has been selected, the ATM interactive application processes the request. At last, the withdrawal is finalized.

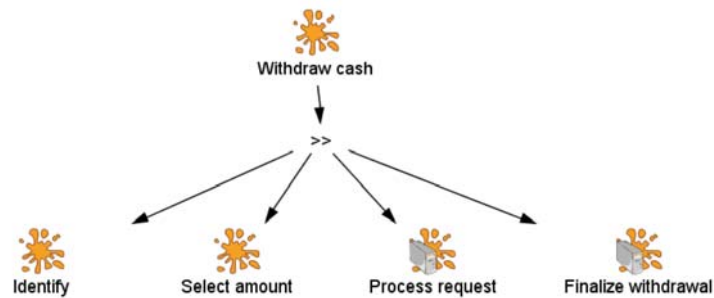


Figure 2. 28 Initial task model of the “Withdraw cash” task

2.5.2 Refinement of the model

Once the main steps (corresponding to sub-goals) have been identified, it is possible to refine each of them, in order to analyze and describe:

- Which precise actions the user has to perform with the interactive system,
- What data s/he has to manipulate

Figure 2. 29 presents the refinement of the sub-goal “Identify”. It is decomposed in two tasks that have to be performed in a sequential way: “Insert card” and “Insert code”. Task “Insert card” is refined into three actions:

- “Put card in card slot” depicted by a motor task
- “Insert card” depicted by an interactive input task
- “Ask for PIN code” depicted by an interactive output task

Task “Insert code” (in Figure 2. 29) is refined into three actions:

- “Enter PIN digit 1” depicted by an interactive input task. The user has to know the information “PIN digit 1” in order to be able to perform the task.
- “Enter PIN digit 2” depicted by an interactive input task. The user has to know the information “PIN digit 2” in order to be able to perform the task.
- “Enter PIN digit 3” depicted by an interactive input task. The user has to know the information “PIN digit 3” in order to be able to perform the task.
- “Enter PIN digit 4” depicted by an interactive input task. The user has to know the information “PIN digit 4” in order to be able to perform the task.
- “Validate” depicted by an interactive input task

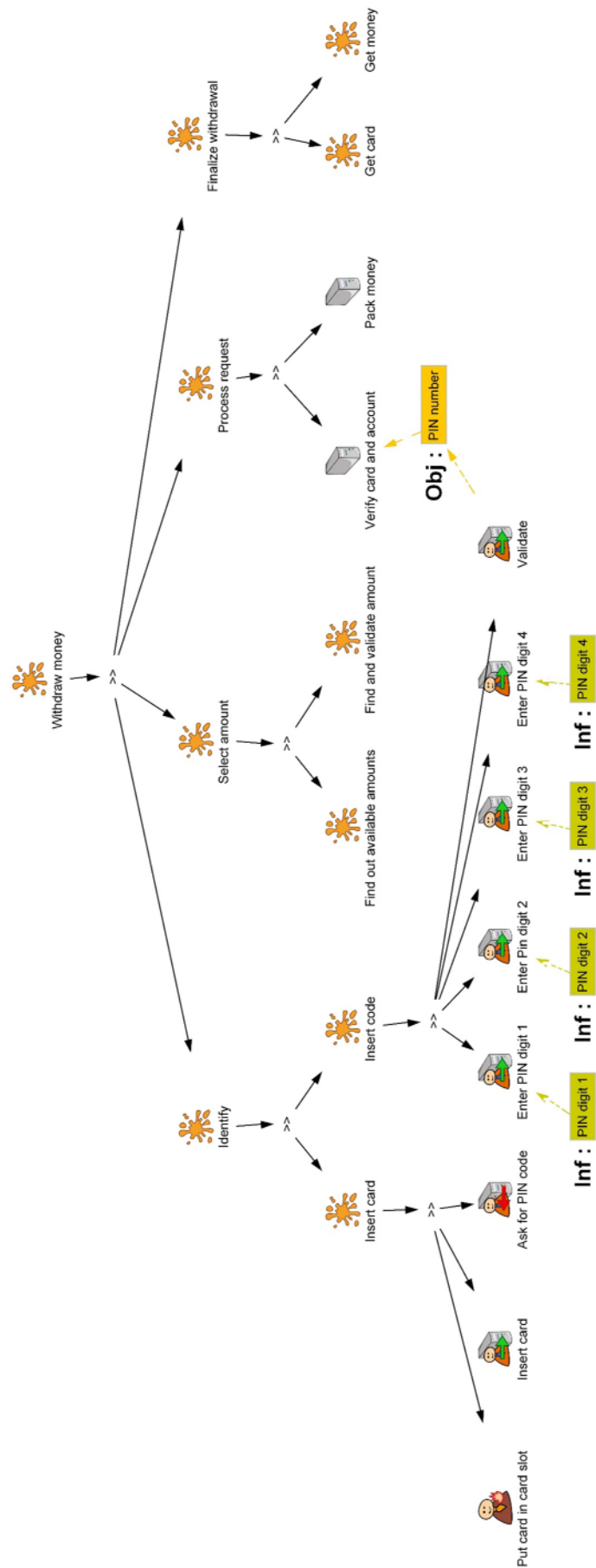


Figure 2. 29 Refinement of the "Withdraw cash" task

2.5.3 Structuring

As the refinement of the task model tend to increase the size and to make to model less legible, subroutines can be created, such as the “Finalize withdrawal” subroutine (in Figure 2. 30).

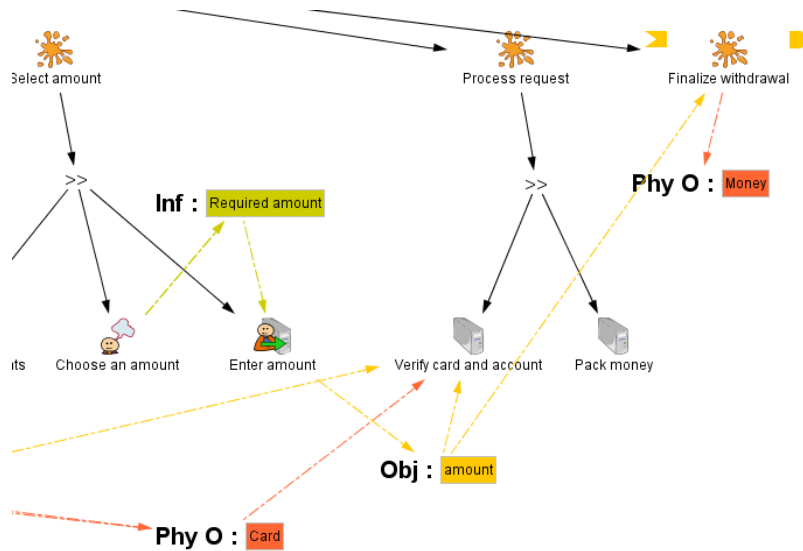


Figure 2. 30 Creation of a subroutine to refine task “Finalize withdrawal”

The description of the subroutine “Finalize withdrawal” is depicted in Figure 2. 31. First, the system displays a message indicating the user that s/he can take her/his card and at the same time the system releases the user credit card (interactive output tasks “Display “take your card”” and “Release card” under the “Concurrent” temporal operator. Then, the user takes her/his card (interactive input/output task “Take card”). The user puts her/his card in a wallet (motor task “Put card in wallet”). Then, the system displays a message indicating the user that s/he can take the money and at the same time it releases the money (interactive output tasks “Display “take your money”” and “Release money”). At last, the user takes the money. In this task model, we can see that the system releases the card thanks to the input/output device “I/O D: card slot”. And that the user takes the card from this input/output device “I/O D: card slot”. In the same way, the system releases the money thanks to the input/output device “I/O D: money slot”. And that the user takes the money from this input/output device “I/O D: money slot”. The interactive output task “Release money” produces the “money” physical object (“Phy O: Money” in Figure 2. 31).

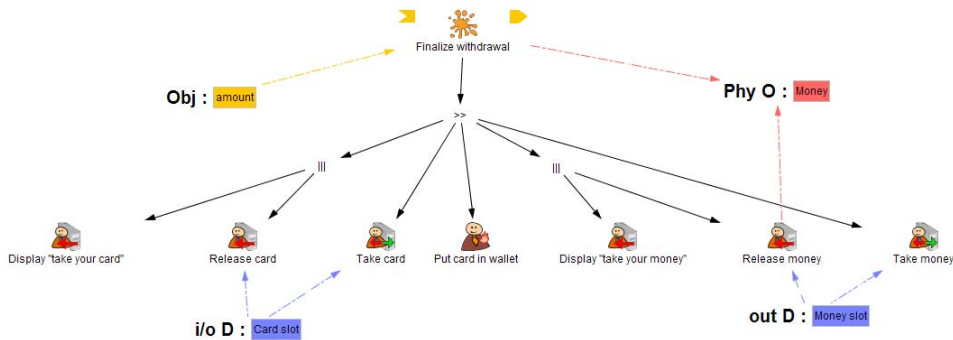


Figure 2. 31 Description of the subroutine “Finalize withdrawal”

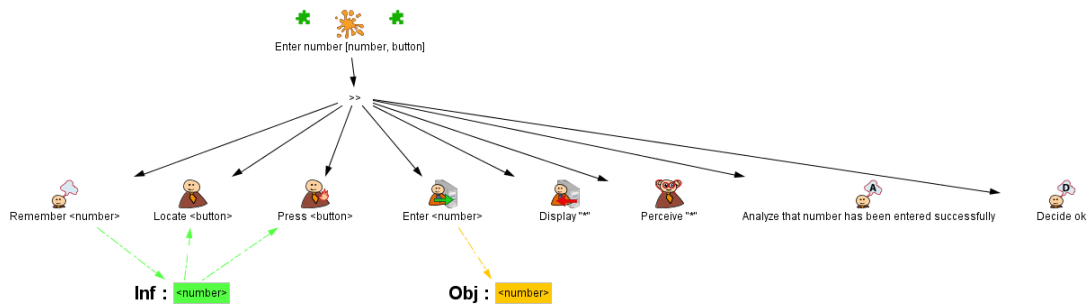


Figure 2.33 Description of the generic component “Enter number”

As the sequence of actions to enter a digit is the same for each digit of the PIN code, whatever the pushed button, a generic component is created and instantiated for the task of entering each digit of the PIN code. The instantiated components are introduced in the main task model, as depicted in Figure 2.32. The description of the generic component is depicted in Figure 2.33.

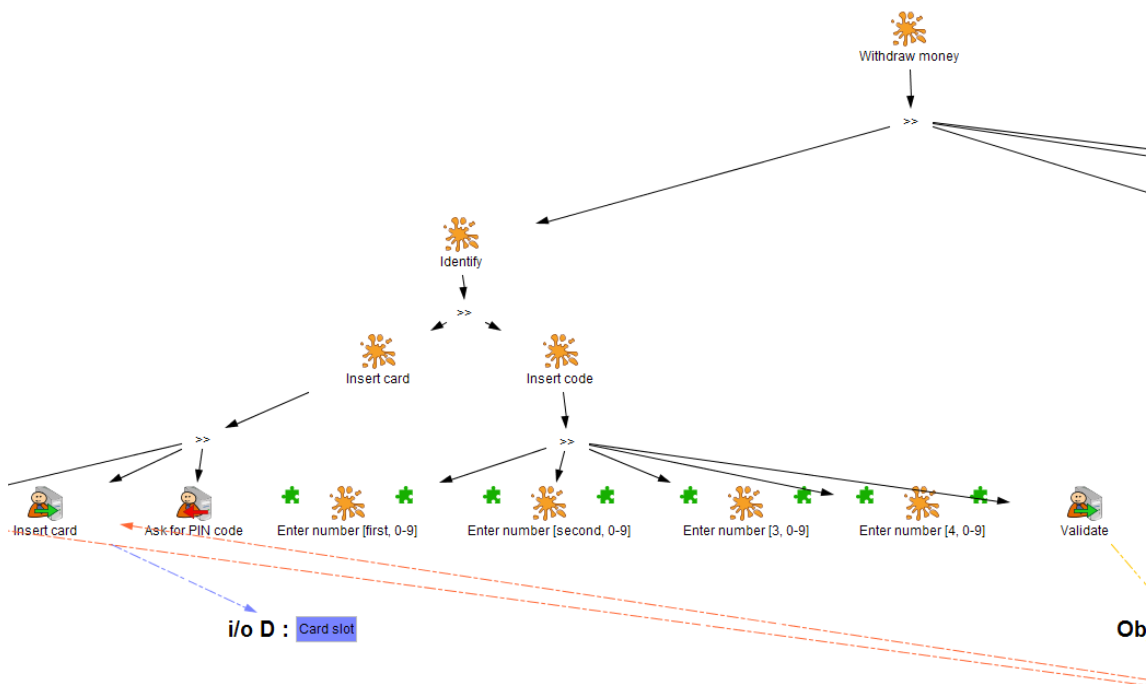


Figure 2.32 Excerpt of the new version of the model embedding instances of generic components

2.6 Beyond this tutorial: advanced design and fine tuning of interaction supported by task models

HAMSTERS notation and tool have been used in several research projects for different purposes in the area of interactive system design and development. The following paragraphs provides an overview on the way task modeling can provide support to interactive design and development activities. We decided not to include those aspects in

the tutorial that was focused on the notation and the tool and targeted at remaining at the introductory level.

2.6.1 Automation, task allocation and task migration

Designing interactive computing systems in such a way that as much functions as possible are automated has been the driving direction of research and engineering both in aviation and in computer science for many years. In the 80's many studies (e.g. [Palmer 1995] related to the notion of mode confusion) have demonstrated that fully automated systems are out of the grasp of current technologies and that additionally migrating functions [Boy 1998] from the operator to the system might have disastrous impact on safety and usability and operationality of systems. Allocating functions to an operator or automating them, raises issues that require a complete understanding of both operations to be carried out by the operator and the behavior of the interactive system.

Tasks models are meant to describe goals, tasks and actions to be performed by the operator while system models represent the entire behavior of the interactive system. Tasks models and systems models thus represent two different views of the same world: one or several users interacting with a computing system in order to achieve their goals. These two views can be integrated at the model level and additionally at the tool level [Barboni et al. 2010]. Such representations can also support the assessment of alternative design options for automation. In [Martinie, Palanque, Barboni et al. 2011], we show the task and system models of two design iterations of an interactive application. These models are analyzed in order to identify potential candidates for automation. The point is to demonstrate that notations supporting a clear dichotomy between user's tasks and system functions make it possible to represent in a complete and unambiguous way allocation of function and tasks migrations.

2.6.2 Dealing with human errors

As user errors are not part of a user goal they are usually omitted from tasks descriptions. However, in the field of Human Reliability Assessment, task descriptions (including task models) are central artefacts for the analysis of human errors. Several methods (such as HET [Stanton et al. 2010] and CREAM [Hollnagel 1998]) require task models in order to systematically analyze all the potential errors and deviations that may occur. However, during this systematic analysis, potential human errors are gathered and recorded separately and not connected to the task models. Such non integration brings issues such as completeness (i.e. ensuring that all the potential human errors have been identified) or combined errors identification (i.e. identifying deviations resulting from a combination of errors). We argue that representing human errors explicitly and systematically within task models contributes to the design and evaluation of error-tolerant interactive system. However, existing task modeling notations, even those used in the methods mentioned above, do not have a sufficient expressive power to allow systematic and precise description of potential human errors. Based on the analysis of existing human error classifications, we proposed several extensions to existing task modelling techniques to represent explicitly all the types of human error and to support their systematic task-based identification [Fahssi et al. 2015]. These extensions are integrated within the CASE tool HAMSTERS.

2.6.3 Performance evaluation

Task models can be used to predict user performance, qualitative and quantitative, while accomplishing tasks. For example, from a task model it is possible to determine the combination of tasks that will lead the user to reach her/his goal. In these possible combinations, it is also possible to determine the shortest path as well as the longest path. From a data perspective, using HAMSTERS task models makes it possible to produce statistics on required information and knowledge to accomplish a task.

Quantitative performance estimation can be done once the estimated minimum and maximum time properties have been filled in for each task in the models. In this way, it is also possible to estimate how much time a user will have to remember one or several pieces of information required to accomplish tasks. And then, according to human models, it is possible to determine the probable memory retention rate of the user. For example, in [Fayollas et al. 2014], we demonstrated how task models can be used to assess the impact of dependable computing architectures on the usability of interactive systems. Task models, when associated to other modelling techniques, can also be a key technique to support the assessment of performance variability in socio-technical systems, as presented in [Martinie, Palanque, Ragosta et al. 2013].

2.6.4 Modelling process

Task models can also be at the center of a development process based on models. In [Martinie et al. 2012], we proposed a development process for the design, implementation and evaluation of safety critical interactive systems. It deals explicitly with requirements that target interactive critical systems (requirements for the system and requirements for the development process). Beyond that it integrates the training program within the process providing a unique opportunity to deliver timely and with a perfect match both a system and its training material. It also explicitly describes the articulation between high-level Safety Integrity Levels (SIL) and low-level ones and thus provides integration for formal and informal approaches.

2.7. Conclusion

This tutorial has presented the HAMSTERS notation and its associated tool. Each aspect of the notation has been introduced covering task types, operators and collaborative activities involving groups of users. These concepts have been exemplified on a cash machine case study which is easy to grasp and remains of a reasonable size for such a tutorial. However, HAMSTERS has been used in large industrial context from various domains such as aeronautics, space, web and entertainment. Experience gained from these projects is always used for improving HAMSTERS which remains a notation and a tool under development.

Beyond the aspects presented in section 2.6, we are working on extending the tool to support the analysts in scenarios identification and management which is a critical element of user centered approaches as discussed in the introduction.

References

Anderson R., Carroll J., Grudin J., McGrew L., Scapin D. Task analysis: The oft missing step in the development of computer-human interfaces; its desirable nature, value, and role. INTERACT 1990: 1051-1054

- Anett, J. Hierarchical Task Analysis. In Diaper Dan, Stanton Neville (Eds), *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 67-82). Lawrence Erlbaum Associates, 2004.
- Barboni E., Ladry J-F., Navarre D., Palanque P., Winckler M. Beyond Modelling: An Integrated Environment Supporting Co-Execution of Tasks and Systems Models. In *Proc. of EICS '10*. ACM, 143-152.
- Boy G. Cognitive Function Analysis for Human-Centered Automation of Safety-Critical Systems. *Proceedings of ACM CHI 1998*: 265-272
- Calvary, G., Coutaz, J., Nigay, L. From single-user architectural design to PAC*: a generic software architecture model for CSCW. In *Proc. of CHI '97*. ACM, 242-249.
- Caffiau S., Scapin D., Girard P., Baron M., Jambon F. Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. *Interacting with Computers* 22(6): 569-593 (2010)
- Cockton, G., & Woolrych, A. (2001). Understanding inspection methods: Lessons from an assessment of heuristic evaluation. *People & Computers*, Springer, pp. 171–192
- Diaper, D. (2004). Understanding task analysis for Human-Computer Interaction. *The handbook of task analysis for human-computer interaction*. Lawrence Erlbaum Associates.
- Ellis C. A., Gibbs S. J., Rein G., Groupware: some issues and experiences, *Comm. of the ACM*, v.34 n.1, p.39-58, Jan. 1991.
- Fahssi, R., Martinie, C., Palanque, P. Enhanced Task Modelling for Systematic Identification and Explicit Representation of Human Errors. *IFIP TC 13 INTERACT conference*, (2015), LNCS 9299, part IV, Springer Verlag.
- Fayollas, C., Martinie, C., Palanque, P., Deleris, Y., Fabre, J.-C., Navarre, D. An Approach for Assessing the Impact of Dependability on Usability: Application to Interactive Cockpits. *EDCC 2014*: 198-209.
- Forbrig P., Martinie C., Palanque P., Winckler M., Fahssi R. Rapid Task-Models Development Using Sub-models, Sub-routines and Generic Components. *IFIP conf. on Human-Centric Software Eng., HCSE 2014*: 144-163
- Gong, R. & Elkerton, J. (1990). Designing minimal documentation using the GOMS model: A usability evaluation of an engineering approach. *CHI 90 Proc.ACM DL*
- Greenberg, S. Working through Task-Centered System Design. In Diaper, D. and Stanton, N. (Eds) *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 49-66). Lawrence Erlbaum Associates. 2004.
- Hollnagel E. *Cognitive reliability and error analysis method (CREAM)*. Elsevier, 1998.
- Kieras, D. GOMS Models for Task Analysis. *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 83-116). Lawrence Erlbaum Associates, 2004.
- Martinie C., Palanque P., Barboni E., Winckler M., Ragosta M., Pasquini A., Lanzi P. Formal tasks and systems models as a tool for specifying and assessing automation designs. *ATACCS 2011 conference*: 50-59, ACM DL.

- Martinie C., Palanque P., Navarre D., Winckler M. and Poupart E. Model-Based Training: An Approach Supporting Operability of Critical Interactive Systems: Application to Satellite Ground Segments, EICS 2011, pp. 141-151, ACM DL
- Martinie C., Palanque P., Barboni E., Ragosta M. Task-Model Based Assessment of Automation Levels: Application to Space Ground Segments. Proc. of the IEEE SMC, Anchorage, 2011
- Martinie, C., Palanque, P., Navarre, D., Barboni, E. A Development Process for Usable Large Scale Interactive Critical Systems: Application to Satellite Ground Segments. HCSE 2012: 72-93.
- Martinie C., Palanque P., Winckler M. Structuring and Composition Mechanisms to Address Scalability Issues in Task Models. IFIP TC 13 INTERACT conference, (2011) 589-609, Springer Verlag.
- Martinie C., Palanque P., Ragosta M., Fahssi R. Extending procedural task models by systematic explicit integration of objects, knowledge and information. Europ. Conf. on Cognitive Ergonomics, 2013: 23-34, ACM DL.
- Martinie, C., Palanque, P., Ragosta, M., Suján, M.-A., Navarre, D., Pasquini, A. Understanding Functional Resonance through a Federation of Models: Preliminary Findings of an Avionics Case Study. SAFECOMP 2013: 216-227.
- Martinie C., Navarre D., Palanque P., Fayollas C. A generic tool-supported framework for coupling task models and interactive applications. EICS 2015: 244-253
- McGrath J. E. Groups: Interaction and Performance. Prentice Hall, Inc., Englewood Cliffs, 1984.
- Mori, G., Paternó, F., Santoro, C. CTTE : Support for Developing and Analyzing Task Models for Interactive System Design. TOSE Journal, 2002, 28(8), 797-813.
- Meyer, D. E., Annett, J., Duncan, K. D. Task analysis and training design. Journal of Occupational Psychology, 41 (1967).
- Navarre, D., Palanque, P., Bastide R., Paternó F., Santoro, C. A tool suite for integrating task and system models through scenarios. DSV-IS'2001; LNCS 2220. Springer; 2001
- O'Donnell, R. D.; Eggemeier, F. T. Workload Assessment Methodology; In Handbook of Perception and Human Performance (Vol. II Cognitive Processes and Performance, pp. 42-41 - 42-49). Wiley & Sons, 1986
- Object Management Group (2003). Unified Modeling Language (UML) 2.0 Superstructure Specification, August 2003. Ptc/03-08-02, pp. 455-510.
- Palanque P., Bastide R., Sengès V. Validating interactive system design through the verification of formal task and system models. EHCI 1995: 189-212
- Palanque P., Basnyat S. Task Patterns for Taking Into Account in an Efficient and Systematic Way Both Standard and Erroneous User Behaviours. IFIP Working Conference on Human Error, Safety and Systems Development, 2004. Springer.
- Palanque P., Martinie, C. Contextual Help for Supporting Critical Systems' Operators: Application to Space Ground Segments Activity in Context Workshop, AAAI conference on Artificial Intelligence, 2011.

- Palmer, E. "Oops, it didn't arm." - A Case Study of Two Automation Surprises. 8th International Symposium on Aviation Psychology, Ohio State University, (1995).
- Pangoli S., Paternò F. Automatic Generation of Task-Oriented Help. ACM Symposium on UIST 1995: 181-187
- Parasuraman, R.; Sheridan, T.B.; Wickens, C.D. A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans*, IEEE Trans. on, vol.30, no.3, pp.286-297, May 2000.
- Paternò, F. *Model-Based Design and Evaluation of Interactive Application*. Springer . 1999
- Paternò F., Mancini C. Developing task models from informal scenarios. *CHI Extended Abstracts 1999*: 228-229
- Paterno, F., Santoro, C. Preventing user errors by systematic analysis of deviations from the system task model. 2002, *Int. Journal on Human Computing Systems*, Elsevier, vol. 56, n. 2, pp. 225-245.
- Paternò F., Breedvelt-Schouten I., de Koning N. *Deriving Presentations from Task Models*. IFIP WG 2.7 EHCI 1998: 319-337, Chapman et Hall.
- Pinelle, D., Gutwin, C., and Greenberg, S. Task Analysis for Groupware Usability Evaluation: Modeling Shared-Workspace Tasks with the Mechanics of Collaboration. *ToCHI*, 2003, 10(4), 281-311
- Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In C. E. O'Malley (Ed.), *Computer-supported collaborative learning* (pp. 69-197).
- Rosson, M. B., & Carroll, J. M. (2001). *Usability engineering: scenario-based development of human-computer interaction*. Elsevier
- Sinnig D., Chalin P., Khendek F. Use case and task models: An integrated development methodology and its formal foundation. *ACM TSEM*. 22(3): 27 (2013)
- Stanton, N.A., Harris, D., Salmon, P.M., Demagalski, J., Marshall, A., Waldmann, T. Dekker, S., and Young, M.S. Predicting design induced error in the cockpit, *Journal of Aeronautics, Astronautics and Aviation, Series A*, 2010, 42(1), 001–010.
- Swearngin A., Cohen M., John B., Bellamy R. Human performance regression testing. *IEEE international conference on Software Engineering, ICSE 2013*: 152-161
- van Welie, M., van der Veer, G.C. Groupware task analysis. *Handbook of Cognitive Task Design*, LEA, NJ (2003), pp. 447–476.
- Wilson S., Johnson P. Bridging the Generation Gap: From Work Tasks to User Interface Designs. *CADUI 1996*: 77-94
- Wilson S., Johnson P., Kelly C., Cunningham J. and Markopoulos P. Beyond hacking: A model based approach to user interface design, In *Proceedings of HCI'93*, 217—23, University Press, BCS HCI