Interacting with Computers 22 (2010) 569–593

Contents lists available at ScienceDirect

Interacting with Computers

journal homepage: www.elsevier.com/locate/intcom

Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models

Sybille Caffiau [a,b,*], Dominique Scapin [b], Patrick Girard [a], Mickaël Baron [a], Francis Jambon [c]

[a] LISI, ENSMA, Téléport 2, 1 avenue Clément Ader, 86961 Futuroscope Cedex, France
[b] INRIA, Domaine de Voluceau, Rocquencourt, B.P.105, 78153, Le Chesnay, France
[c] LIG/MultiCom, University of Grenoble, Bâtiment C, B.P. 53, 38041 Grenoble Cedex 9, France

ARTICLE INFO

Article history:
Received 26 June 2009
Received in revised form 3 June 2010
Accepted 9 June 2010
Available online 19 June 2010

Keywords:
Task models
Tool-supported task modelling
Empirical assessment

ABSTRACT

Task analysis is a critical step in the design process of interactive systems. The large set of task models available today may lead to the assumption that this step is well supported. However, very few task models are tool-supported. And in this latter category, few of them are based on a clear semantics (in this article, the word semantics is used with the following definition: "the meaning of a word, phrase, sentence, or text" from Compact Oxford English Dictionary®). This paper focuses on tool-supported task models and provides an assessment of the features that have been considered as essential in task modelling. It compares the different tool-supported methods, and evaluates the actual use of these features in K-MADe, a tool aimed at contributing to the incorporation of ergonomics into the design process of interactive systems through activity and task analysis. The originality of the K-MADe tool is to be based on a model whose expressive power lies on computable syntax while trying to be usable by every modelling knowledge designer. This facilitates task description and analysis, but also model query and the migration within software engineering models and software lifecycle steps. Evaluation results demonstrate the usefulness of an increased expressive power for task models, and their acceptance by users. They also enlighten some weaknesses in the K-MAD method and suggest further improvements.

1. Introduction

The benefits that well-designed interfaces bring to users are largely acknowledged today (Shneiderman and Plaisant, 2009). Across domains (life-critical, industrial, home and entertainment, collaborative,...) and beyond the functional point of view, it is becoming more and more important to take the usability point of view into account. Effectiveness, efficiency and user satisfaction must be considered early in system design in order to increase the acceptance of interactive applications. In a user-centered approach, studying the user activity – also known as task modelling or task analysis – may be used to support user needs (Norman and Draper, 1986; Dix, 1991). User-centered criteria such as task conformance and task coverage (Dix, 1991) have been elicited. Consequently, task analysis has acquired more and more value, together with the enrichment of task models and analysis methods.

Initially developed by ergonomists and usability specialists to evaluate user activity, task modelling is currently often included in software design process studies. Several contributions have tried to link task models to software models such as UML (e.g. Reffye et al., 1988). One can assume that task analysis could be part of global requirement analysis and could thus be considered to be a more precise description of some parts of user requirements as parts of task models are clearly defined. Attempts have also been made to generate parts of software from task models (Mori et al., 2004). Dynamic aspects of task models could also be described in process modelling techniques, such as BPMN[1] for example.

However, this success is mainly limited to research, and task models are not extensively used in actual software design. As for UML notation several years ago, task modelling suffers from a relative lack of reliable tools. Following the definition of Minsky (Minsky, 1988), a model of something is useful when it helps their users in resolving questions they ask themselves about that thing. In the meantime, problem solving requires the model semantics to be clear. One way to reach this point is to be able to use tools that enforce the model rules. Ensuring consistency of models, allowing dynamic exploration such as simulation, giving the opportunity to query and compare designs, facilitating links with other models, all these are things a tool might do, founded on a non-ambiguous

* Corresponding author at: LISI, ENSMA, Téléport 2, 1 avenue Clément Ader, 86961 Futuroscope Cedex, France. Tel.: +33 5 49 41 13 16.
E-mail addresses: sybille.caffiau@ensma.fr (S. Caffiau), Dominique.Scapin@inria.fr (D. Scapin), girard@ensma.fr (P. Girard), baron@ensma.fr (M. Baron), Francis.Jambon@imag.fr (F. Jambon).

[1] http://www.bpmn.org/.

0953-5438/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.intcom.2010.06.003

semantics it might exploit, to allow design verification and validation.[2]

Unfortunately, among the relatively high number of studies on task modelling, very few tool-supported task models have been designed, and only one has tried to exploit a well-described component semantics (Paterno, 2004). Furthermore, when tools do exist, they generally do not implement the whole model.

This study is a part of a more global work on interactive application design based on task-based approaches. Our goal in this paper is to evaluate the usefulness of task model features for users involved in interactive system design. Based on our assumption that tools are necessary for using models, we first selected the most complete available tools, and made up both (1) a systematic and feature based comparison of tool-supported systems, and (2) an empirical assessment of the chosen model and tool.

This paper is organized as follows: Section 2 analyzes the existing task models that are tool-supported, in order to highlight their strengths and weaknesses. In Section 3, the K-MAD (Lucquiaud, 2005) model and the K-MADe tool (K-MADe, 2006) are briefly described, and compared with the other methods and tools, pointing out the different concepts that are really implemented in the tool. Sections 4 and 5 report the evaluation of the method. This evaluation is divided into three parts: (1) an observation of the presence of the concepts in relation with the weaknesses reported in Section 2, (2) a case study conducted on several applications in order to put the tool in practice, and (3) an empirical study where a particular attention is paid on actual usage of new features. Then, the last section discusses the main results and prospects of this research.

2. Comparative study of tool-supported task models

The (descriptive) representations/notations used for analysis are usually identical to the (technical) method output representations/notations. Some methods have a translation step so that they can output different representations that may be used in a software engineering method, such as those supported by the UML notation (Object Management Group, 1997).

One aspect of HTA (Annett, 2004; Annett and Duncan, 1967; Shepherd, 1989; Shepherd, 1995) that makes it a rather simple task analysis method is that it has only one single analysis representation. Most task analysis methods have a relatively small number of steps and representational forms, such as Scenario-Based Design (Carroll, 2000) and ConcurTaskTrees (Paterno, 2004). The coverage of all analysis steps (as TOOD Abed et al., 2004; Mahfoudhi et al., 2001) implies an important learning cost to use it. Methods such as GOMS (John and Kieras, 1996a,b; Kieras, 2004) and UAN (Hix and Hartson, 1993) are partial task analysis methods in that they deal only with the later steps of what a task analyst does.

In software engineering, different steps use one or more representations (Diaper, 2004). For instance, Data Flow Diagrams (DFD), Entity Life Histories (ELHs), Entity Relationship Diagrams (ERDs), and Process Outlines (POs) are used in Structured Systems Analysis and Design Methodology (SSADM), in the same way as Use Cases, Activity Diagrams and Interaction Diagrams are representations used in UML. It would therefore seem advisable for task analysis output representations to be in the form of one or more of such software engineering representations or to be easily translated into them. However, many task analysis methods (Balbo et al., 2004) have been developed by researchers with a psychological background and focused mainly on the first steps of user interface design, so their outputs often do not integrate well with those from software engineering.

These task analysis methods generally have a very wide coverage and usually share common features (Limbourg and Vanderdonckt, 2004). Many methods (input–output diagrams, functional flow charts, Petri nets, graphs, etc.) are available; some of them make it possible to describe and formalize tasks carried out with software tools (for instance Limbourg and Vanderdonckt, 2004; Scapin and Bastien, 2001; Paternò et al., 1997). Beside expressive power, validity and usability, essential issues concern the paths between the various models of analysis, design, and development, the used languages, as well as the associated tools and services.

Two major kinds of classifications have been made among task methods. In the first one, Balbo and coworkers (Balbo et al., 2004; Jambon et al., 2001) propose a comparison to help designers to choose a notation adapted to their needs. From the study and the use of several task models, they propose a taxonomy that follows six topics: goal, usability for communication, usability for modelling, adaptability, coverage and extensibility. The second classification follows a very different approach. The comparison of task model components may lead to building a meta-model. Limbourg has followed this approach in Limbourg and Vanderdonckt (2004), and has compared many models through their meta-model. This work thus aims at identifying common concepts in order to connect them together. A meta-model approach allows the communication and transformation between task models (to exchange data, for example (Limbourg and Vanderdonckt, 2004). In relation to the large range of models these two approaches demonstrate how large the task modelling domain is. However, their features have not been evaluated in any detail yet.

In this paper we also propose a comparison of concepts of some task models. We did not use a meta-model based approach because we concentrated on the semantics each model expresses. We preferred discussing each concept, in order to be sure that their meaning in each model could be compared. Recently, we used a meta-model approach in another work (Caffiau, 2009).

In order to improve knowledge exchanges between persons with different skills that contribute in the interactive application design, task analysis proposes to express data around tasks. In order to perform task analysis, designers use several notations. Some of them are "generic" ones: they do not address task analysis specifically but they may be used for task analysis (BPMN BPMN 2.0, 2010). As they do not address task analysis, they cannot be used to perform any verification on the task analysis semantics.

Some models represent a *technical* viewpoint on tasks and activities. For example, the Tibco business studio tool was developed for business and IT users (Tibco Business Studio, 2010).

In the user-centered design approach, one important issue is the communication between domain experts (users of the designed application) and designers (developers, human factor specialists...). Some other models aim at bridging the gap between both expert types. We consider two different approaches. Firstly, several tools allow designers to derive information from the task model. For example, with this objective in mind, GOMS (John and Kieras, 1996b) proposes a temporal analysis of activities (according to expressed tasks, designers can determine execution time for a scenario). Secondly, some models are developed as communication support of the application design to improve user-designer exchanges. This paper presents a study of the last task model type. It proposes a study on the structure of task models developed to improve user-designer communication in a user-centered application design approach. As our goal is to carry out a practical evaluation, we decided to focus on tool-supported task models. This approach may be viewed as limited. It is nevertheless the most pragmatic, accurate and reproducible approach we could use. Among the large

[2] We use in this paper the classical software engineering (adapted from [IEEE-STD-610]) distinction between verification (verify that a clearly specified problem is solved properly) and validation (ensure that what is done meets the actual requirements). Verification is generally made by semi-automatic or automatic tools, while validation requires human interpretation.

number of task model languages proposed in literature, few of them have a clear semantics. In this category, articles do not usually describe the complete semantics and syntax of the models. As a consequence, some of their characteristics are ambiguous. Moreover, their authors have gradually improved most of these task models, and their accurate comparison requires a huge version tracking effort: their semantics usually vary according to publications.

Conversely, tool-supported task models, with or without explicit definition of their semantics, can be assessed, including through user testing. These tools could thus be considered as the "iron bird" of the semantics of the model. The tools are instant snapshots of the semantics of their models at a tangible moment in the evolution of these models. To summarize, tools have both the advantage of freezing the semantics of the models, and of accurately expressing the semantics of these models.

Moreover, participants of this evaluation are university students (from the University of Poitiers). Due to this evaluation constraint, we selected task model tools with the following constraints:

- *Graphical*: assuming that a graphical interface improves the use
- *Documented*: by user manual (such as K-MADe K-MADe, 2006), video tutorial (such as IBM IBM Task Modeler, 2010), modelling examples (such as TKS TKS, 2010 model for ADEPT (Johnson and Johnson, 1993)
- *Free and downloadable: we* did not include some tools presented in some papers (Dittmar et al., 2005; Gamboa and Scaplin, 1997; Biere et al., 1999; Heinrich et al., 2007; Molina et al., 2005) when we did not find the website to download it or if the tool was not free (such as the taskArchitect tool (TaskArchitec, 2010) which is free only to express 20 tasks at most).

We found five candidates[3] for testing: CTT (Paternò, 2001) (CTTE), Diane+ Tarby and Barthet, 2001 (TAMOT), GTA Van der Veer, 1996 (EUTERPE), an Eclipse plug-in (IBM task modeler) IBM Task Modeler, 2010 and a task analysis environment (AMBOSS) AMBOSS, 2008 that can be used for task model design.

2.1. Rapid description of tool-supported task models

We will briefly present task models and their associated tools prior to comparing them in the next section.

2.1.1. CTT (CTTE)

ConcurTaskTree (CTT) is defined by its authors as a notation for task model specifications to overcome limitations of notations used to design interactive applications. Its main purpose is to provide an easy-to-use notation, which represents different temporal relations at the same abstraction level and which can be used by a task modelling novice. CTT is based on a hierarchical structure of tasks represented by a tree-like structure. Each task is displayed in a graphical tree-like format. Temporal relationships mainly derive from LOTOS operators (ISO Systems, 1984; Paternò and Faconti, 1992) which are able to express temporal relationships among tasks at the same abstraction level.

ConcurTaskTree Environment (CTTE) is its associated tool, which is largely used in the academic area. CTTE handles two concepts: *tasks* and *objects*.

2.1.2. Diane+ (TAMOT)

Diane+ is a full methodology, which attempts at bridging the gap between software engineering and human factors. As CTT, Diane+ focuses on the design of interactive applications. Diane+ integrates some characteristics from general ergonomics and cognitive psychology and also intervenes in the software development life cycle. Moreover, Diane+ supplies a specific model called Diane+ H for the task analysis representation.

Diane+ is a hierarchical task model composed of a set of operators. The representation is different from the CTT representation: Diane+ uses boxes to specify task sub-levels. TAMOT is its associated tool. Diane + H handles only one concept: *tasks*.

2.1.3. GTA (EUTERPE)

Groupware Task Analysis (GTA) is a task analysis method that is intended for modelling complex environments where many people interact with interactive systems. Two tools are associated with this model: GTA tool (no longer supported) and Euterpe. This last tool does not include any pre-defined scheduling function. GTA handles five concepts: *tasks*, *roles*, *agents*, *objects* and *events*.

2.1.4. IBM task modeler (IAWB)

The IBM Task Modeler (named since 2006 as IBM information Architecture Workbench (IAWB)) is an Eclipse-based tool. It is used for modelling human activities as hierarchical task trees. It can be used to produce several task analysis diagrams such as hierarchical task analysis (HTA) models or role and goal (RAG) diagrams. We studied its use to perform hierarchical task analysis models. IAWB was developed for creating "expressive diagrams that inform and engage design stakeholders" (IBM Task Modeler, 2010). IAWB handles three concepts: *tasks, objects and roles*.

2.1.5. AMBOSS

We did not find papers on the task model for the AMBOSS tool (Giese et al., 2008). We therefore have considered models and tools as a whole and we have sometimes inferred the model from the tool. The AMBOSS tool aims at assisting task analysts in designing their task models especially for safety critical systems. It provides several views dedicated to specific aspects.

The tasks are hierarchically composed of sub-tasks according to scheduling decomposition operators. AMBOSS handles four concepts: *tasks*, *objects*, *roles* and *barriers*. Barriers are introduced to express the protective mechanism of tasks. For example, in order to complete an identification task, a barrier may express a condition on the validity of the entered password (for example: creation of a barrier "password_OK"). During the simulation of the task model, the designer indicates if the barrier is triggered or not (the task is performed only if the barrier is triggered).

2.1.6. Discussion about models and tools

Prior to comparing these five tools in detail, two remarks must be made. Whilst every model is hierarchical, the task decomposition is not the same. The differences are presented below. In addition, by using these tools, differences can be identified for concept definition despite the vocabulary they use. These differences seem to be due to both the different origins of the models and to the goal of their designers. Even if the terms used in the tools are not the same, we considered what they represent independently from their terminology. Finally, we observed some distance between models and their implementation in tools. In the remainder of this paper, we study only the models from the point of view of their tools by only considering the implemented models (in the tools) as the used models.

2.2. Comparing expressiveness and querying of task model through their tools

In this section, we present a comparison of the notations and tools selected above: CTT (CTTE), Diane+ (TAMOT), GTA (Euterpe),

[3] Some of them, like TOOD, were not selected for evaluation as they were no longer supported.

IBM information Architecture Workbench (IAWB) and AMBOSS. Our final goal was to evaluate the expressivity of the models in real use. Two points were of great importance here: (1) the semantics of these models was not explicitly given by their authors, so, it was sometimes difficult to find out the actual meaning of parts of the models; (2) we chose models that own a graphical tool, which is available to use in an evaluation stage. So, when the notation semantics was not clear, we interpreted it from the tool implementation. We compared two main points of the models: the *expressive power* of each of their components and their *querying power*.

2.2.1. Expressive power

The expressive power defines the capability of the model to express user activity. Expressed activity may be used as part of user requirements, as the starting point of user-centered application design. This capability is assessed through the syntax of task models. The comparison of the five tools: CTTE, TAMOT, Euterpe, IAWB and AMBOSS highlights four different components: *task, objects, events* and *users*. The expressive power of each of these components is compared for all five tools.

2.2.1.1. Expressive power of tasks. Tasks are defined by several components. Each of them plays a particular role. We identified three different roles. The first one is task definition (*information and characteristics*). The two others specify the scheduling. The scheduling characteristics follow two levels. First, the scheduling at the task level (*local scheduling*) is composed of characteristics that concern the task itself. Secondly, the scheduling at the task model level is composed of scheduling characteristics with implications on several tasks (*global scheduling*). For each category a list of every characteristic is given and their presence/absence in each task model are stated.

2.2.1.1.1. Task information and characteristics. This category groups two different entities: the *informational task attributes* (attributes that add information about the task, but which do not characterize it) and the *task characteristics* (except for scheduling characteristics). The *informational task attributes* are: the **name**, the **number** (unique identifier of the task), any related **remark**, the **goal** (or objective), a **media support** and the **room** (place where the task may be performed). The first six rows of Table 1 detail these informal task attributes for each tool. The set of task characteristics is composed of the **executant** (who performs the task), the **significance** (the importance of the task), the **frequency** (how many times the task is usually performed) and the **platform** (what specific target the task is performed on). One can notice that *informational task attributes* are very subjective: no precise definition is given for them. In contrast, the *characteristics* have usually strong expressed semantics. Attribute values are often pre-defined in the tools, which allows attribute computing. The models identify these characteristics by using different words even if they represent the same concept. For example, the place where the task can be performed is named *Room* in AMBOSS and *Location* in IAWB. This vocabulary difference is explained by the situation of the initial model creation (Limbourg and Venderdonckt, 2004). The second part of Table 1 describes for each tool: the name used to identify the characteristics, when it differs from the concept name (in *Italic*); and the pre-defined values.

2.2.1.1.2. Discussion: the expressive power of task information and characteristics. The five tools define as common task data: the **name**, the related **remark** and the **executor**. While the name and the remarks are expressed using non-constrained texts (String), the task executor is defined from existing values in the tools. IAWB expresses task executors by two components: the *type* (Cognitive, Communication, Physical, Other) and the *Allocation* (User, System). Even if the five tools use different terms to represent the different executors, we can define four executor types. When the task executor is known, the task can be performed by a human (*Human*), by the system (*System*) or as a result of the reaction of the system to human actions (*Interactive*). The last executor type is the type used when task executor is not yet defined (*Composed*). The existing executors may be different for each tool (see Table 2).

Interactive tasks (tasks triggered by user action and producing system reaction) are not described in the AMBOSS task model. Its tasks are composed of abstract tasks or task that are performed by human or system only. These executor types stand for a low abstraction level of activity description.

In Euterpe, a task may be created without any executor. Moreover, even if it is possible to define whether a task is composed or elementary, this tool does not distinguish tasks performed by humans or by the system. Finally, when a task is interactive, the interaction can be indicated from the interactions defined by the designer. In IAWB, tasks are not specified as composed task. However, a task type may be "other", which can be used to express a non elementary task.

Some tools add the possibility to provide additional comments about tasks with the addition of the **goal** of tasks (CTTE, Euterpe and IAWB), a **media support** path (Euterpe) or the **room** where the task is performed (IAWB and AMBOSS). Moreover, some characteristics are also added: the **significance** of tasks (Euterpe and IAWB), the **frequency** (CTTE and IAWB) and the **platforms** on which the task is performed (CTTE). Addition of frequency and platform characteristics in CTTE allows the use of CTT task models

Table 1
Expressive power of task information and characteristic.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Name	String	String	String	String	String
Number	–	–	–	Integer (automatically done)	–
Remark	String	String	String	String	String
Goal	String	–	String	Three Strings (Goals, Inhibitors, Motivation)	–
Media	–	–	String Path	–	–
Room	–	–	–	Locations (composed by strings)	String
Executor	Category: user, system, interaction, abstract	Style: manual, automatic, interactive	Type: complex, unit (user or system), interaction, ø	Type: Cognitive, Communication, Physical, Other Allocation: User, System	Role: human, system
Significance	–	–	Relevance: normal, critical, dangerous, optional, ø	Business critical (Boolean), Importance (five value scale), Safety critical (Boolean)	–
Frequency	High, medium, low, ø	–	–	Integer (five value scale)	–
Platform	PDA, Desktop, Mobile, Voice	–	–	–	–

Table 2
Executor types of the task model tools.

	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Interactive	√	√	√	√	–
System	√	√	√	√	√
Human	√	√	√	√ (Cognitive, Physical)	√
Composed	√	–	√	–	√

to generate interactive interface prototypes (Paternò and Santoro, 2002). Unlike CTTE, we do not find any information on the use of these added characteristics for the other two tools (Euterpe and IAWB) except on their informative role.

This task information and these characteristics constitute task description. No rules are established to define the task information and characteristics during the task modelling process. For example, for frequency characteristics, designers have to choose the frequency level from a set. However, the choice of the value depends on the designer's appraisal. Moreover, few characteristics are defined from pre-defined value sets (semantically understandable by systems) and when they are, their semantics is not always clear, which shows that semantics is not exploited by the tool. For example, what is the significance of a task defined by system allocation and cognitive type? These two reasons (lack of definition rules and non-use of semantics) limit the use of characteristics by the tools.

Task models are used to create interactive applications and also to help ergonomics specialists to detect dysfunctions (Balbo et al., 2004). Task models thus ought to integrate concepts and notions traditionally used in ergonomics. However, the existing task model tools are closer to the HCI specialist needs than to those of ergonomics specialists (Couix, 2007). For example, only the IAWB tool introduces the task **number** as an attribute of tasks, which is used to organize the tasks.

2.2.1.2. Local scheduling. The local scheduling attributes are the set of task characteristics that influence the task scheduling of the task independently of its sub-tasks. From the use of the tools, we identified six characteristics in this set: the **duration** of the task execution, the task **optionality**, the **interruptibility**, the **iteration** and the **cooperation**.

The **duration** task attribute gets temporal information concerning task execution. We split time information into two main types. Firstly, duration indication (how long has task been performed?) and secondly, an indication about the task execution time (when does the task execution start? When does it finish?).

The **optionality** of a task indicates whether the task has to be performed for the activity to be complete or not (then, it is an optional task). A task can be optional for different reasons (task not mandatory in the system process, task required only under some conditions...) and its execution is dependent from user's preferences (for example: completing an optional field in a form) or process requirements (for example: adding paper to print is mandatory only if the printer has run out of paper).

Defining a task as **interruptible** indicates that its execution can be stopped or not.

In order to notify that some tasks can be performed several times, the **iteration** can be present as a task characteristic.

The last local scheduling attribute found in the study of tools is the **cooperation** task attribute. This attribute is a part of the local task scheduling because, in order to be performed, a collaborative task requires all actors to be available.

In Table 3, for each tool, we indicate the type or the values used to express these characteristics.

2.2.1.2.1. Discussion: expressive power of local task scheduling. Only CTTE does not allow the definition of the **duration** using a time measure. AMBOSS provides duration expression as a quadruple (numbers of days, hours, minutes and seconds) and IAWB has only one value of days or hours or milliseconds or minutes or months or seconds or weeks or years. In CTTE and AMBOSS the duration is composed of three attributes (in order to specify the duration coverage (minimum, maximum, average) or the task place in the activity scheduling (the start point, the finish point and the duration of the task). However, the duration attribute seems to be linked more to a particular scenario than to a description of activities.

CTTE indicates the **interruptibility** of the tasks, using operators for CTTE. In CTTE, a task is interruptible if it can be suspended when one of its sister-tasks is performed. Then, this tool considers that interruptions are performed at pre-defined moments (unlike phone call events for example).

Only two tools take into account the task **iteration**: CTTE and TAMOT. While CTTE allows the addition of the iteration characteristic to a task, the number of iterations cannot be specified. TAMOT allows indicating the minimal and maximal number of task executions.

Last, in CTTE, tasks may be defined as **collaborative** (i.e. several actors can perform the task). This characteristic is necessary in the CTTE task description because the CTT model recommends designing one model for each actor of each activity. Then, the collaborative task attribute allows linking the different actors modelled for the same activity.

2.2.1.3. Global scheduling. This category concerns only the scheduling operators. Prior to analyzing the expressive power of scheduling operators (Table 4), we specify how they are used in the tools. While CTTE, Euterpe, IAWB and AMBOSS decompose their tasks into sub-tasks, thus building task trees, TAMOT uses boxes of tasks. Moreover, only CTTE uses operators to link the sister tasks whereas the other tools use parent–child links (decomposition operators).

In order to express the global scheduling, we identified four operators and two types of interruptions. Interruption can be definitive or not: interrupted tasks can continue their execution once the interrupting task is completed (*without cancelling*) or can be definitely interrupted (*with cancelling*). The operators are: *sequence* (tasks are performed by analyst defined order), *concurrence* (the tasks are performed concurrently and can be performed at the same time), *no order* (the tasks are performed in any order, every task will be performed and only one task can run at a given time) and *choice* (only one task is performed). Table 4 summarizes the presence of scheduling operators in each tool.

Table 3
Expressive power of the task local scheduling.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Duration	3 integers (min, average, max)	–	–	1 integer	3x4 integers (duration, min, max) (H, D, M, S)
Optionality	Boolean	Boolean	Not clear	Not clear	–
Interruptibility	Operator	–	–	–	–
Iteration	Boolean	Integer (min, max)	–	–	–
Cooperatible	Boolean	–	–	–	–

Table 4
Expressive power of scheduling operators.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Sequence	√	√	–	√	√
Concurrent	√	√	–	√	√
No order	√	√	–	√	√
Choice	√	√	–	√	√
Desactivation	√	–	–	–	–
Interruption	√	–	–	–	–

2.2.1.3.1. Discussion: expressive power of global task scheduling. In CTTE, some other scheduling operators allow the specifying of information exchange. However, this indication is an informative element that does not modify the global scheduling (two tasks linked by the concurrent-with-information-exchange operator have the same behaviour in the CTTE simulation tool as two tasks linked by the concurrent operator). Therefore, we do not distinguish between both types of scheduling operators in our comparison. In IAWB, some other scheduling operators are defined. These operators are the four main variations. They express the possibility or not to perform all sub-tasks (that express optionality).

Scheduling operators do not exist in the Euterpe model. The links between tasks only indicate that a task is composed of other tasks. Scheduling operators may be defined through a label on the links between tasks. Due to this definition type, the tool cannot interpret them and then, cannot exploit their semantics.

CTTE, TAMOT, IAWB and AMBOSS allow scheduling tasks **sequentially**, **concurrently**, without **any order** or according to a **choice**. In addition to the choice operator defined as the execution of only one task from the subtask set, IAWB has operators that allow for the execution of several sub-tasks (the number is not specified). Only CTTE takes into account the **interruptions** (*deactivation* or *interruption*) by using scheduling operators. These operators allow the specification of the interruptibility of tasks.

Finally, the use of some scheduling operators and task characteristics can create an ambiguity. For example, what is the significance of an optional task that is performed only if it is selected (alternative scheduling)? In (Paternò, 2001), Paternò indicates the ambiguity problems that can be raised using the CTT task model scheduling operators.

2.2.1.4. Expressive power of objects. In addition to tasks, other elements have been considered as essential in task models (Dix, 2008): **users**, **events** and **handled objects** (objects on which user operations apply through functions, devices, tools, etc.), which increase the task model expressive power. **Users** are the humans who play a role in the activity. They perform the tasks or are at the origin of the task (as would a supervisor). While the activity progresses, some events may interfere. They are part of the activity. The last concepts are the objects handled in the activity. The different task model tools incorporate some of these concepts.

2.2.1.4.1. CTTE. CTTE handled objects (Paterno, 2004) are task properties. They are characterized by: a **name** (string); a "**class**", to be chosen among *string, numeric, object, description* or *position*; a **type** among *perceivable* (object presenting information or allowing action of users) and *application* (internal to the system); an **access mode** (*only reading* or *modification*); a **cardinality** among *low*, *median* and *high*; and the **platforms** where the object is represented. To our knowledge, no documentation describes the concepts of class and cardinality in detail. According to our use of the CTTE tool, we associate the need of the cardinality characteristic with the generation of interfaces (Paternò and Santoro, 2002) based on CTT diagrams. However, the CTTE simulation tool does not take objects into account and we only found some documentation relating to the use of objects for interface generation. CTTE does not include users and events as specific concepts.

2.2.1.4.2. TAMOT[4]. The task model Diane+ integrates handled objects, named **data**, and uses them to define **conditions**. However, in the associated tool, TAMOT does not allow the defining of *data* and the only editable condition is the pre-condition, expressed as strings.

2.2.1.4.3. EUTERPE. In EUTERPE, handled objects are first-class components. They are characterized by a **name** (string); a **list of attributes** (each attribute is composed of a **name** (string) and a **value** (string)); and a **list of users** (the users who can manipulate the object). The users are defined as **labeled agents**. Relations can be defined between agents and objects (*owner, create, destroy, use/inspect, change*). An **agent** is defined by a **name** (string); a **type** (*individual, organization* or *computer system*); and a **role** (a *set of tasks* performed by an agent). Moreover, EUTERPE allows the definition of **events**. These are composed of a **name** (string) and the **set of tasks** linked with (task set).

2.2.1.4.4. IAWB. As for CTTE, the IAWB tool defines objects as task attributes. They are defined by a name (a String), a description (a String) and a type (a String). Values cannot be associated with tasks and can be composed neither of attributes nor of other objects.

In addition to handled objects, IAWB tasks can be associated with **roles**. Roles in IAWB are users that perform the task. All components associated to tasks are defined in a model dictionary and can then be "linked" to several tasks. No semantics is defined to these links.

2.2.1.4.5. AMBOSS. The AMBOSS environment allows defining handled objects and the association between tasks and objects; and defining users. The handled objects are composed of a **name** (string), its **description** (string) and its **type** (physical or information). Each object can contain other objects (definition of object composed of objects).

One of the main usage of objects is to express pre and post-conditions. Table 6 presents the definition of these attributes in the tools. All the models include the pre-conditions associated to the tasks. While CTTE, TAMOT and Euterpe express these pre-conditions as strings, AMBOSS and IAWB express pre-conditions as a necessary presence of components (barriers and messages for AMBOSS and keywords for IAWB). However, both elements are expressed as strings themselves. In order to specify the consequences of a task execution on the handled objects, Euterpe and IAWB include post-condition expressions. These pre and post-conditions are therefore expressed as strings without any link with objects and without any possibility to take the object values into account.

2.2.1.4.6. Discussion: the expressive power of objects. Table 5 presents the expressive power of objects in the five task model tools. With the exception of TAMOT (which does not contain *data* from the Diane+ model), all the tools contain the concept of objects. They can be divided into two categories. First, the tools that consider objects as task attributes (as CTTE and IAWB), and second the tools that consider objects as first class model components (as in Euterpe and AMBOSS). The definition of objects as task attributes requires that handled objects are transferred to other tasks by using operators in order to perform several tasks.

In CTTE, a particular object attribute is its cardinality. It is used to help the designer to define the interactive element that represents that object. Moreover, perceivable objects may be a table or a window and so this tool associates interactive objects to tasks. The introduction of these elements (cardinalities and perceivable objects) in the task model illustrates the link between handled

[4] http://www.ict.csiro.au/staff/Cecile.Paris/IIT-Track-Record-Past-Projects/Projects/Isolde/Tamot/Index.htm.

Table 5
Expressive Power of Objects.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Abstract Object	–	–	–	√	√
Concrete Object	√	–	√	–	–
Attribute name	Not computable	–	√	–	√
Attribute type	Not computable	–	–	–	Not computable
Attribute value	Not computable	–	√	–	–
Users	Task owner	–	√	√	√
Events	–	–	√	–	–

Table 6
Expressive Power of task attributes that can use objects.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Pre-condition	String	String	String	Keywords	Barriers + messages
Post-condition	–	–	String	Keywords	–

objects and interface presentation. This definition is close to a system-based point of view whereas in all other tools, object concepts try to focus on ergonomic aspects. While objects defined in CTTE are concrete (a value is associated to the object as soon as it is defined), IAWB and AMBOSS do not allow giving a value to object attributes thus staying at an abstract level. This abstract level of definition freezes the manipulation of task model objects, which limits the expression to a static state of the world.

The condition verifications are mandatory for task execution. Then, to allow the validation of task models by the user and thus the validation of the task scheduling (using simulation), these conditions need to be computed. In order to compute the conditions, the definitions of used entities and conditions have to be computable. However, all tools define conditions using non-computed strings and do not manipulate the defined objects. The same observation can be made concerning post-conditions when they are present (in Euterpe and IAWB). Only AMBOSS integrates the checking of the pre-conditions prior to performing the tasks. However, these pre-conditions are expressed by the presence of *barriers* and *messages* (as strings).

Only Euterpe considers **events** as objects. Except for TAMOT, all tools take into account the notion of **users** but they do so in different ways. In CTTE, a tree is designed for each role (for example, a CTT tree for the salesman, one for buyer) with a global tree that contains all the activities (here, the sales activity). Then there is a tree for the organization and a tree for each actor. However, this modelling stops potential data exchanges (such as those of the handled objects) between several users and limits the description of group activities. IAWB considers the user as a task attribute and thereby does not make any distinction between user data and task data. However, there are two different types of data that may have consequences on another. For example, a task actor has a function on a group. This function may justify the presence of a task that modifies object values. So, user information and handled objects are linked but not dependent. On the contrary, Euterpe identifies two different concepts: the roles and the agents. Lastly, in AMBOSS, several humans and systems can be created as specific concepts and associated to the tasks. The definition of users as a concept independent from the tasks allows associating a same user to all the tasks performed by that user.

2.2.2. Model querying

Querying models in a tool relates to its capability to use the semantics of the model according to its syntax. We identified two different features: the verification of the model and its simulation. The capability of model querying guarantees that a model is designed according to its associated tool. The model syntax allows the use of task models by other approaches in order to generate interfaces (such as Mori et al., 2003), to allow migration between different platforms (such as Calvary et al., 2001) or exchange data between task models expressed by different syntaxes (Limbourg et al., 2001). We consider that these features do not depend on task model tools. They rely to independent tools that may use designed task models as input. As an example, the tool TERESA (Mori et al., 2003) uses CTT task models (designed by using CTTE) independently from CTTE itself, for generating user interfaces.

2.2.2.1. Verification. Three tools: CTTE, Euterpe and AMBOSS, allow the verification of the grammatical correctness of the models. However, none allows the verification of every model entity. All verify the grammatical consistency of the decomposition of the task (hierarchically), indicating for example whether an interactive task consists of tasks performed by other executors. The tools CTTE and AMBOSS allow the grammatical verification of the used operator. This type of verification enables to establish the coherence of the operators and their use. For example, tools detect the absence of scheduling operators whereas tasks are non elementary (AMBOSS) or have some sister tasks (CTTE).

As shown in the previous sections, CTTE, Euterpe, IAWB and AMBOSS incorporate objects that complete task composition. However, only Euterpe allows a grammatical verification of these elements. It verifies two types of object constraints: the verification of the *cardinality constraints* and the verification of the *type constraints* (Fig. 1).

The *cardinality constraint* is composed of the set of constraints of coherence between the defined entities and the used ones. For example, Euterpe detects the presence of events that do not trigger any task or the presence of agents without any role.

The *type constraint* detects the inconsistencies in the type definition of entities. For example it can detect an object composed of itself.

Lastly, even if all the tools allow the description of conditions (as pre/post-conditions), none of them trigger any verification on conditions: they do not manage their semantics. The types of grammatical verification for each tool are summarized in Table 7.

2.2.2.2. Simulation process. The last capability that we studied is the simulation (Table 8). Owing to the limited number of computable features in the tools, only two, CTTE and AMBOSS, allow model simulation. This observation raises the issue of using model components and their validation. Some models are clearly designed to edit task models and facilitate the communication (as Euterpe which does not pre-define scheduling operators). But why do other tools have some pre-defined components that are not used?

CTTE and AMBOSS simulate models according to the scheduling specified by operators. Then the produced scenarios may be recorded and replayed. A scenario is a particular running example of task model (a specific sequencing of tasks). However, as objects are not manipulated by tasks by using references on entities (as

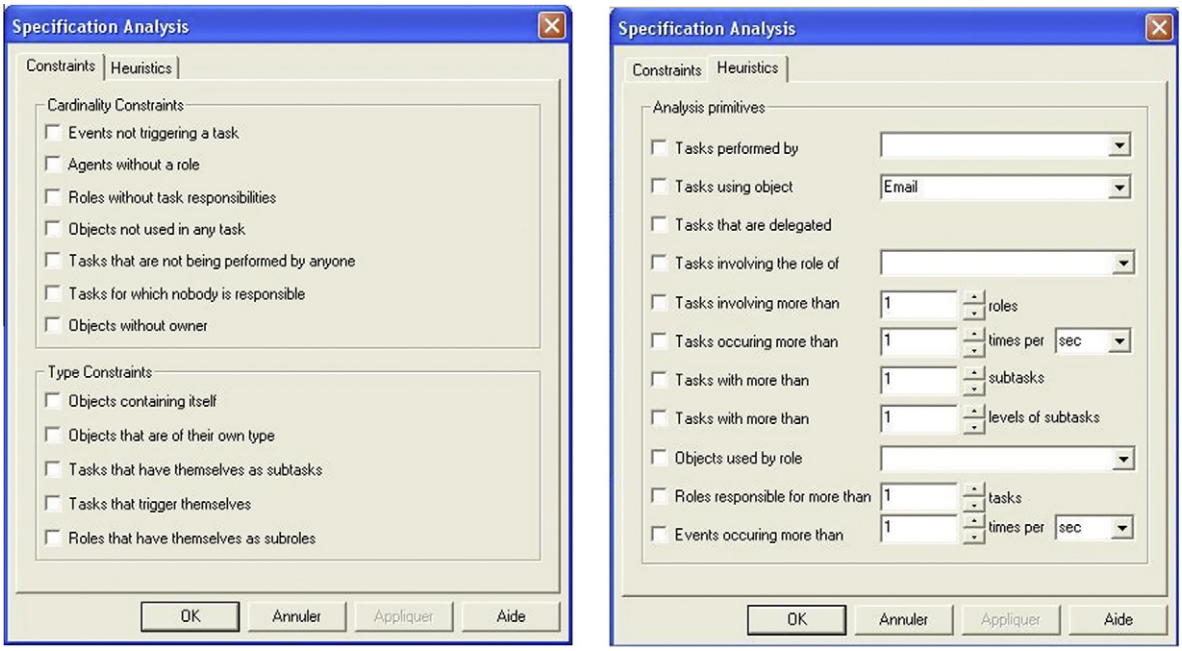

Fig. 1. Verification of (a) constraints and (b) heuristics on the Euterpe models.

Table 7
Verifications of tools.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Hierarchical	√	–	√	–	√
Operator	√	–	–	–	√
Entity	–	–	√	–	–
Expression	–	–	–	–	–

Table 8
Simulation presence in tools.

Tools	CTTE	TAMOT	Euterpe	IAWB	AMBOSS
Scheduling	√	–	–	–	√
Object	–	–	–	–	–
Expression	–	–	–	–	√
Record/replay scenario	√	–	–	–	√

TaoSpec Dittmar et al., 2005 textually does, for example), no simulation of the state of the objects can be performed. AMBOSS takes the conditions of the task execution into account. These conditions are composed of the presence of barriers and messages expressed as strings. For example, the entry of a password to use an application may be expressed as a barrier.

2.3. Main issues from model comparison

From the above study, several main weaknesses can be pointed out about the five tools considered, from the point of view of their potential usability in the interactive application design process.

The first observation that results from the use of tools is the difference between model compositions and features implemented in tools. For example, the task model Diane+ includes objects whereas its associated tool (TAMOT) does not manage these elements. The task models are defined from requirement studies; thus, the elements that compose the task models are elements that have been proved to be a part of model activity expressions. The absence of these elements in their associated tools limits the expressive power of task models.

Moreover, a precise **semantics** is often lacking in task model description. Some components are just given an informal description, and are represented in the model as a free text or a simple list of values. In addition, each characteristic can have a value, but how to use this value is not clearly defined. For example, the importance levels of tasks in IAWB are not semantically described, so one cannot know how to actually use it. Does it mean that a task that is important may interrupt another task? This point limits the use of some task model components for full modelling – as the user does not know how and why they can be used.

As we indicated above, task model components can be divided into two different types: the pre-defined components (components for which values are defined from a pre-defined set) and the informational ones. **Few components are defined from pre-defined value sets (and then are computable)**; the most commonly defined component is the scheduling operator. Except for Euterpe, all of the tools pre-define its values. Pre-defined components can be used to perform verification (computing them) on task models using the simulation tool for example as in CTTE and AMBOSS. While some components are modelled to inform, others ought to be used to animate models, as objects. The definition of objects following a specific syntax can allow the specifying of information exchanged between tasks and to perform some computing (object value modifications...).

In addition to an informational definition of task model components (as task name), we observed that **conditions are also textually defined**. The definition of conditions should complete the scheduling of activities expressed by scheduling operators. Nevertheless, their textual definition does not allow their verification and consequently, their usage. Consequently, the conditions become only informative.

The simulation of models provides a dynamic view. This requires computable characteristics, and the evolution in the state of the "activity world". However, we noticed that **the objects are not taken into account in the dynamics of the model**. This could be explained by both previous observations. As the complete activity description implies object definition (Dix, 2008), the simulation of activity description implies the dynamic simulation of object

evolution. By simulating task models that integrate objects, one can obtain a simulation that is closer to the dynamic behaviour of the future system. Therefore, the addition of dynamic objects increases the set of elements that can be verified.

Lastly, task models can be used as documents about activities that allow for example discussions between several users in order to compare different task models. This characteristic of task models is limited because **the data and its semantics are not accessible**.

These six points limit the capability for these models to be actually used. A new task model, K-MAD, was specially designed to address these limitations. This model aims at incorporating the concepts presented above and to remove the ambiguities pointed out. An associated tool (K-MADe) has been jointly developed.

3. Presentation of K-MAD and K-MADe

The initial work concerning task modelling for user interfaces (Scapin, 1988) had several objectives: to consider the way in which users represent their task, and not only the logic of the data-processing, nor the "prescribed" task; to take into account the conceptual and semantic aspects and not only the syntactic and lexical aspects; to obtain a structuring of the tasks in a uniform way; to carry out descriptions from a declarative (current state of things) as well as procedural (way to reach these states) point of view; to allow parallelism and not only sequential representation (synchronization of the tasks); and to be computable.

From these first ideas and various requirements, a first task analysis model ("MAD": Méthode Analytique de Description) was proposed (Scapin and Pierret-Golbreich, 1989a; Scapin and Pierret-Golbreich, 1989b) at the intersection between ergonomics, computer science, and artificial intelligence. The tool E[5]MAD was developed (Delouis and Pierret, 1991).

In parallel, a method for task information gathering was defined (Sebillotte, 1991); it consists mainly of semi-directed interviews based on the "Why? and How?" technique. A practical methodology of task analysis for extracting the relevant characteristics useful for the design of interfaces was proposed (Sebillotte and Scapin, 1994; Sebillotte, 1994).

This work was continued according to an iterative process: development (for example, implementation of the model in an object oriented fashion), organization of the tasks at various levels of abstraction, on site validation (house automation, management of ship fires, air traffic control, railway regulation, etc.), modifications, and validation again.

Research work (TKS, 2010; Gamboa, 1998) also focused on the design and specification of interfaces. Two models were designed: a task model (MAD* TKS, 2010) and a model for conceptual interface design (ICS Hammouche, 1995) as well as procedures for transforming one model into another, and links with low level interface aspects (via the ILOG-Views©[6] toolbox). The work led to the implementation of the task analysis editor EMAD* and to a software design environment (ALACIE (Gamboa, 1998)).

From this previous work and the study of other task models, a task model kernel was created. This kernel corresponds to the need for a task model that contains the different task model components. The current model (K-MAD), based on the review of the existing models and tools, is described in Lucquiaud (2005). The current version of the tool (K-MADe) is described in Baron et al. (2006), and available at: http://kmade.sourceforge.net/.

[5] "E" for Editor.

[6] http://www.ilog.com/products/views/.

3.1. The K-MAD model

K-MAD intends to facilitate the implementation of a user-centered, model-based approach for the design and the ergonomic evaluation of interactive software based on tasks.

K-MAD can be used either for on-site needs collection, user activity analysis, product validation, etc. It can also be used during the various phases of the development cycle, from task model specification and design to user evaluation and documentation writing phases. K-MAD aims at being a kernel that allows the description of all types of activity that would satisfy the requirements identified in the related work section.

3.1.1. Main characteristics of the model

K-MAD is a hierarchical model. It represents the user's activity as a tree of tasks, from the most general one (root) to the most detailed ones (elementary actions). In order to express activities, the K-MAD model is composed of components and scheduling characteristics.

3.1.2. Components

Four concepts are defined in the K-MAD model: tasks, objects, events and users.

Tasks: a task is defined by a **name**, a **number**, a **goal**, a **duration**, and **feedback** (observable effects by the user). Moreover, some other information can be associated to the task: **observations** (a free textual commentary), level of **importance** (small, average, high), **frequency** (small, average, high), **executor** (user, system, interactive, abstract). When the executor is a user, a specific feature can be defined, the **modality** (sensorimotor, cognitive).

Handled objects: these objects characterize the environment of the user, they are objects the user handles **or** that influence the course of the user's activity. Objects are defined by a **name** and **attributes**. They are represented in K-MADe by two levels of object classes:

- *Abstract objects*: characteristics relating to the concept handled by the user with abstract attributes (characteristics of the abstract object). For example, a car (expressed as an abstract object) may be defined by a color (an abstract attribute).
- *Concrete objects*: they correspond to instances of abstract objects with concrete attributes (that allow a value to be assigned for each characteristic of abstract attributes of abstract objects). For instance, the Smith's car is red (a concrete value of the abstract attribute).

Concrete objects are organized in **groups**, which are defined by their **name** and a type of group.

Users: set of users involved in the described activity. When the task executor is a user, the various users associated with the tasks are identified as actors, characterized by a **name**, a level of **experience** in the modelled activity (beginner, average, expert) and **competence** (skills).

Events: set of events that can be triggered or caused by the described activity. They are defined by a **name** as string.

Dynamic models: in order to obtain a dynamic model, a task is associated with potential side effects: post-conditions (actions resulting from the task, i.e. dynamics of the model objects: modification of the attribute values, creation or removal of objects), events (events can be generated during the execution of the task).

3.1.3. Scheduling characteristics

K-MAD tasks own a set of scheduling mechanisms that are applied to the task, independently from others (local scheduling). The

conditions of the execution are the **pre-conditions** (conditions for the task to be achievable) and the **iteration** conditions (conditions for the repetition of tasks).

Moreover, tasks can have several sub-tasks for which the scheduling is described by: a triggering **event**, a **need** (optional, mandatory), an **interruptibility** flag (yes, no), and a **scheduling operator** (sequential, alternate, no order, parallel). If there is no scheduling operator, when a task is not composed of other tasks, it is defined as elementary. Even if *elementary* is not a scheduling operator, it belongs to this set because it concerns information about the decomposition of the task.

3.2. The tool: K-MADe

K-MADe implements all characteristics of the K-MAD model. It allows the task models to be described, modified and queried (query functionalities are under development). It is available in French and in English and was developed with version 1.5 of the Java JDK. It addresses all kinds of users (users with different backgrounds) wishing to describe all types of activities. K-MADe can also be used to help the end-user in his task via a help system, especially during training. However, this environment was particularly intended for ergonomics and Human Computer Interaction (HCI) specialists.

In addition, depending on their training, the use of current parts of the model and tool can vary: from simple graphical description (for non-computer trained users) to detailed specification of tasks, objects and conditions (for computer trained users).

3.2.1. Technical characteristics

The graphics interface uses the Swing toolbox. Different API were also used:

- JGraph[7] for the modelling part of the task tree.
- L2Fprod[8] for adding graphic components to the basic Swing API.
- JGoodies[9] for the "look and feel" of the tool.

The Express language (ISO EXPRESS, 1994) was used for modelling the task model and for the "state-of-the-world" objects (current state objects). The implementation of Express modelling was carried out by a "custom" API which provides all the services for creating and handling objects.

Finally the XML language was chosen to describe the grammars associated with the expressions of pre-conditions (see Fig. 3a), post-conditions and iterations. This option facilitates the design of tools that can exploit these elements of the model easily. The handling of XML files requires the use of an additional API: DTDParser.[10] This API makes it possible to analyze an XML file according to its DTD.

3.2.2. The K-MADe components

The K-MADe tool has been developed in order to be used by people wishing to describe and analyze human operators or users, in computerized environments or not, in real or simulated, site-based or laboratory-based situations. Although all kinds of profiles can use this tool, it is particularly intended for ergonomists and HCI specialists. As there are many skill variations in such populations, the tool allows different levels of description, from simple graphical drawings to detailed mathematical expressions using several tools:

- Graphic editor of the K-MAD task model. Use of direct manipulation techniques for constructing, handling and cancelling tasks (label 1 in Fig. 2).
- Task characteristics editor (see the list above). The editor has three forms. A basic form with only the main task characteristics, a form which lists all task characteristics in tables (label 2 in Fig. 2), and finally a printable form (summarized) with all task characteristics.
- Editor of abstract objects, users, events and concrete objects. Objects can be added, modified and removed. *Sheets* (label 3 in Fig. 2) allow these object definition spaces to be accessed.
- Tool for expressing the pre-conditions, post-conditions and iterations. A grammar defines condition expression components (Fig. 3a). The tool is able to check whether the grammar of expressions is correct (Fig. 3b).
- Simulator for animating task models.
- Various tools for analyzing task models (statistical, coherence, queries...).
- Tool for printing task trees and task characteristics.

More information about the K-MADe tool may be found in its user manual.[11]

4. Evaluation of the K-MAD/K-MADe expressive power

K-MAD includes computable entities and conditions that cannot be found in other task models. Despite studies that demonstrated the pertinence of adding expressive power to task models through object definitions, it seemed interesting to evaluate how these additions can be used in actual task modelling activity. To be fully completed, such an evaluation needs a tool that supports these model features. K-MADe meets this requirement, as it is close to the model.

The aim of our work is to concretely evaluate how computable aspects of task models are effectively used during task modelling. The study is conducted in three phases. Firstly, from our comparison presented in Section 2, we look at how K-MADe covers the different notions the previous methods/tools manipulate. We particularly pay attention to definitions of computable components and management of these notions, and we focus on how K-MADe responds to the major limitations described in Section 2.3. Secondly, observations from a large case study, where task modelling was used for varying purposes, are provided. Thirdly, an empirical study of K-MADe usage, where the use of advanced features was encouraged, is reported.

4.1. Coverage of the kernel

K-MADe aims at offering a tool with a task model kernel that allows the expression of an activity as detailed as if it was described with all other tools. In order to analyze the coverage of this model, we will indicate what elements can be found in K-MADe and compare them with those found in other models. Tables 9–11 summarize the task **characteristics**, **scheduling** and **objects** found in K-MADe.

4.1.1. Task characteristics

Among the task characteristics found in the five task model tools, only two are not implemented in K-MADe, the **room** (present in AMBOSS) and the **platform** (present in CTTE). Both characteristics are defined in the tool to meet their specific requirements. The room is the space where tasks are performed (Giese et al., 2008) and the platform is used to help the production

[7] http://www.jgraph.com.
[8] http://common.l2fprod.com.
[9] http://www.jgoodies.com.
[10] http://www.wutka.com/dtdparser.html.
[11] http://kmade.sourceforge.net/.

of interface prototypes (Mori et al., 2004). These characteristics do not seem essential for task modelling. Nevertheless, if required by the activity domain, they can be fully defined by the way of K-MADe handled objects (see Section 4.1.3).

The K-MADe task characteristics can be classified in three groups following their informational level. Some characteristics are completely informational: the *informative* ones. We consider as elements in this group: task **name**, task **remark**, **goal** and **media support**. Lower informational level characteristics are task elements that are pre-defined (the user chooses among values) but are not exploited in the tool: the **significance** and the **frequency**. The last group contains the task characteristics that are pre-defined and exploited in K-MADe: the task **number** (generated by the tool) and the task **executor** (this characteristic is checked).

4.1.2. Scheduling

Even if K-MADe allows the association of several users with tasks, as most task models, it does not take into account the task **cooperatibility** aspect as such. In K-MADe, the task **duration** is considered as information, and therefore defined as free text. In contrast, all other scheduling components are defined by a pre-defined value: task **optionality**, **priority**, **interruptibility**, **iteration**, **pre-condition** and **scheduling operators**. These pre-defined descriptions allow these components to be exploited by tools (as for simulation). In K-MADe, the **deactivation** and the **interruption** are not expressed as operators (as in CTTE). Moreover, K-MADe was developed to express the user activities describing the achievements of her/his goals (as state in Sinning et al. (2007), to perform *idealized* task models). As the **deactivation** of task does not allow the achievement of its goal, the **deactivation** is not included in K-MADe.

4.1.3. Objects

K-MADe contains all the object types found in the tool-supported task models: handled objects, events and users. Moreover, each object has an attribute name; an attribute type and an attribute value and both types of objects (abstract and concrete) are **present** in the model. No other tool possesses all these concepts. Objects, users and events are defined as **independent task model components**, as first-class components (instead of CTTE). Next, they are used for defining pre/post-conditions or iteration.

4.1.4. Querying power

4.1.4.1. Verification. K-MADe allows the same verifications as the other tools (CTTE, Euterpe and AMBOSS): **hierarchical**, **operator** and **object** (Table 12). Moreover, as Euterpe, K-MADe focuses its verification on the use and syntax definition of elements. However, while in the verification of Euterpe, objects are requested by the users (using the tool presented in Fig. 1), K-MADe automatically performs verifications on objects prior to authorizing their use (for example, to define conditions).

In addition to these objects, K-MADe computes the defined **expressions** (pre, post and iteration conditions) and verifies their syntax according to objects.

4.1.4.2. Simulation. By using pre-defined values, the description of activities allows the simulation of models taking into account more characteristics. As two other task model simulation tools (CTTE and AMBOSS), the K-MADe simulation tool takes into account the **scheduling** and allows the **recording** and **replaying** of scenarios.

As AMBOSS, it integrates the simulation of **expressions**. However, these expressions are composed of mathematical syntax (operators, functions and computer types (see Fig. 3)). This type of definition was used to allow the computation of expressions that

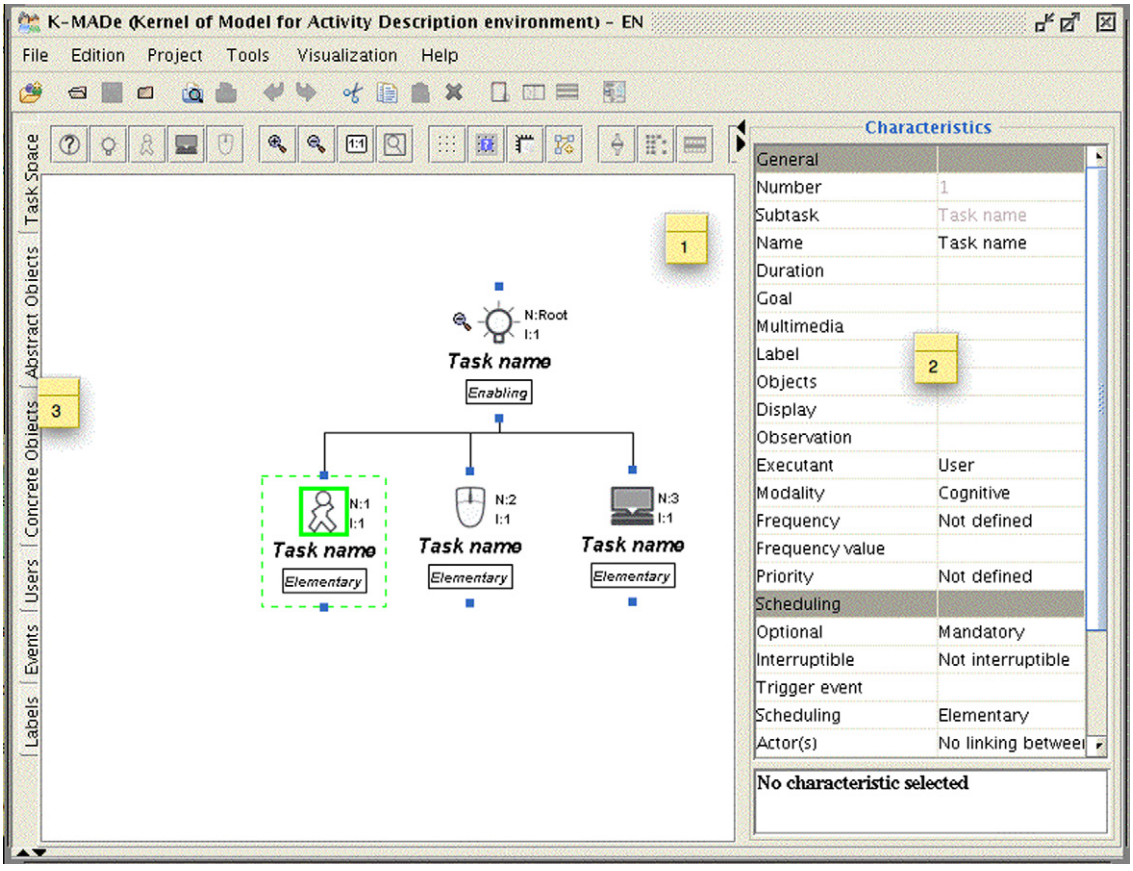

Fig. 2. The main window of K-MADe tool.

a

```
Pre-condition := pre-condition simpleBinaryExpression ı simpleBinaryExpression
simpleBinaryExpression := exp opCompareB exp
opCompareB := < ı > ı <= ı >= ı == ı AND ı OR
exp := date ı function
function := binaryFunction ı unitFunction
binaryFunction := functionNameB groupeName attributName
unitFunction := functionNameU groupeName
data := userEnter ı value
userEnter := string ı integer ı Boolean
value := string ı integer ı Boolean
functionNameB := card ı getValue ı isExistAt ı isExist
functionNameU := card ı getValue ı isEmpty
groupeName := string
attributName := string
```

b

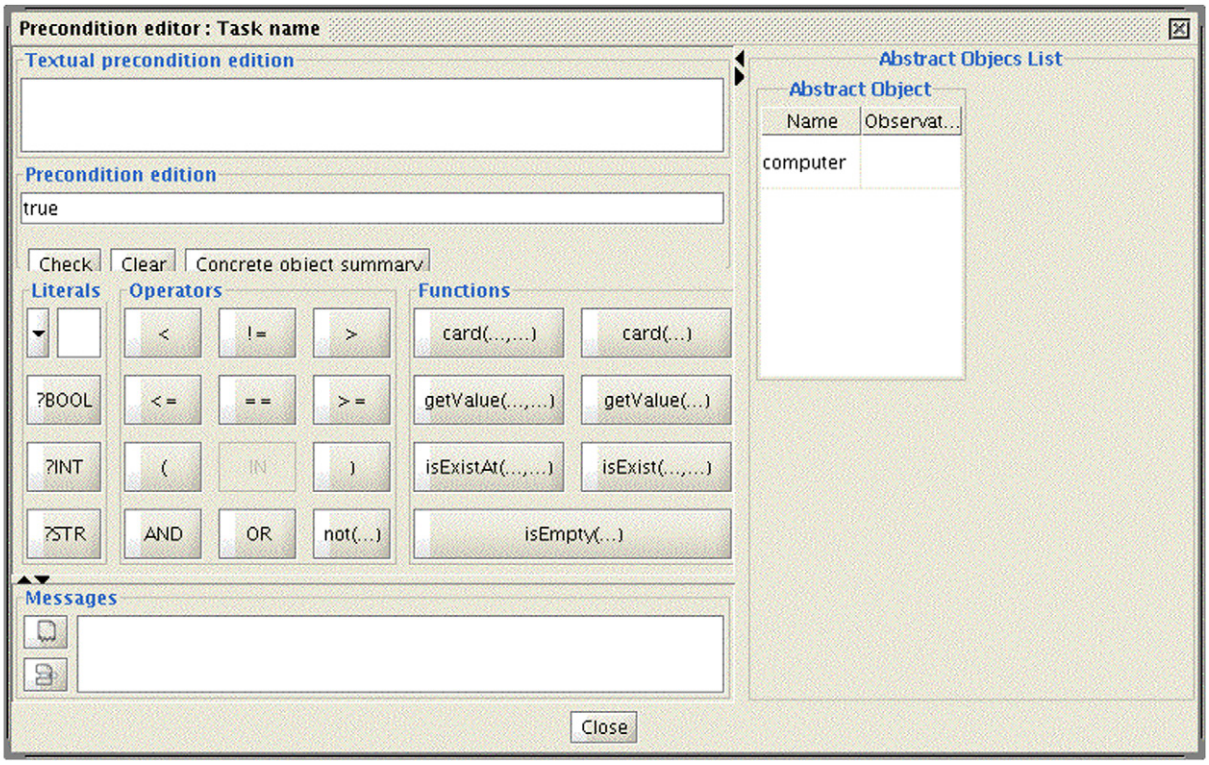

Fig. 3. (a) A part of the pre-condition grammar. (b) The window to write task pre-condition.

Table 9
Task characteristics that can be found in K-MADe.

	Presence	Computable
Name	√ (String)	
Number	√	√ (Automatically generated)
Remark	√ (String)	
Goal	√ (String)	
Media	√ (Path)	
Room	–	
Executor	√ (Interactive, Abstract, User, System)	√
Significance	√ (High, medium, low)	
Frequency	√ (High, medium, low)	
Platform	–	

Table 10
Scheduling elements that can be found in K-MADe.

	Presence	Computable
Duration	√	
Optionality	√ (Boolean)	√
Priority	√ (Low, medium, high)	√
Interruptibiliy	√ (Boolean)	√
Iteration	√ (Formal expression)	√
Pre-condition	√ (Formal expression)	√
Cooperatibity	–	
Sequence	√	√
Concurrent		
No order		
Choice		
Deactivation	–	
Interruption	–	

Table 11
Objects that can be found in K-MADe.

	Presence	Independent
Abstract object	√	√
Concrete object	√	√
Attribute name	√	–
Attribute type	√	–
Attribute value	√	–
User	√	√
Event	√	√

manipulate **objects**. Therefore, the K-MADe simulation tool also simulates the (dynamic) state of the world specified as a set of objects Table 13 shows elements take into account for the dynamic simulation.

As seen above, K-MAD and its tool K-MADe cover the great majority of the concepts that exist in each task modelling tool-supported method. Restrictions are few, and mainly due to a difference in goals: for example, in CTT, the *platform* permits the direct generation of interfaces, which is not considered as a "kernel" function of task models in K-MAD. The next step is to evaluate how useful these features are, and how they solve the restrictions we reported in Section 2.3.

4.2. Case studies evaluation

In order to study the real K-MADe coverage power, we used K-MADe to model five activities. The case studies were chosen to study the modelling of a wide range of interactive applications (interaction type, user number, user type, application platform). Then, we designed the application task models corresponding to activities with different types of users; different system types; and on different platforms. Moreover, task models were used in order to target different goals (Balbo et al., 2004) (design, validation...).

4.2.1. Case study description

We used K-MADe to validate one application (*ParAdmin*) and to design the other four. The choice of these case studies allowed us to design task models in three different contexts. First, three of them (the *webmail*, the *volley-ball marking sheet* and the *Mastermind game*) were designed for training. Second, the *ParAdmin* application was designed in a research context, with frequent evolutions. And last, *Genindexe* was designed in order to produce an operational software application. Table 14 summarizes the characteristics of each case study and why the K-MAD model was used.

4.2.1.1. Webmail. The first application is a classical mail application, in its web form. This design is based on a task model expressing all activities that allow communication by email (address book management, email production, management of received emails...).

4.2.1.2. ParAdmin. In order to study the role of the K-MADe task model to validate an application interface, we modelled the management of a data warehouse. A tool, named *ParAdmin* (Bellatreche et al., 2008), was developed to perform this task without previous task analysis. In order to allow its use by many people, task models were used to propose some changes.

4.2.1.3. Volley-ball game marking sheet. The last studied activity concerning only one user is the completion of a volley-ball game marking sheet (during a game). This activity is traditionally performed using a paper sheet. The goal of the design of this activity is to migrate to a tablet platform and then, to automate calculations and verifications (for example, to evaluate from the data entered whether a game is finished or not).

4.2.1.4. Mastermind. The design of the mastermind game shows the needs for an application used simultaneously by several users. Mastermind is a code-breaking game. A player (the code-maker) defines a code and a second player (the code-breaker) guesses it. This game was developed to be used on computers by one (the computer plays the code-maker role) or two players.

4.2.1.5. Genindexe. The last application was designed for managing the activity of a genetic analysis laboratory.[12] This laboratory must fulfill the rules of regulatory organisations such as the FDA (US Food and Drug Administration) and the COFRAC (French accreditation institution). A task model has been designed to create an application adapted to the activities of all employees according to their security requirements (such as traceability of samples).

4.2.2. What are the main K-MADe benefits?

K-MAD and K-MADe have been designed in order to deal with the weaknesses underlined by the comparison of existing task model tools. Therefore, they try to answer the questions left unsolved by the other systems. In this section, we detail how they solve the questions from Section 2.3. Then, we give a summary of our observations about the expressive power of K-MAD/K-MADe in the context of our case studies. Moreover, these case studies illustrate how K-MAD compensates for some expressive limitations from the use of task models to design interactive applications that Sinning et al. Sinning et al. (2007) have identified. In the last part, we show how K-MAD addresses (or not) these expressive limitations.

4.2.2.1. K-MADe answers to issues from the comparison of models. The comparison of task model tools reveals six points that limit their use to perform task analysis:

Table 12
Verification of the K-MADe tools.

Hierarchical	√
Operator	√
Object	√
Expression	√

Table 13
Simulation presence in K-MADe.

Scheduling	√
Object	√
Expression	√
Record/replay scenario	√

Table 14
Characteristics of case studies performed using K-MAD.

Application Name	Number of users	Platform type	Goal of the use of K-MAD
Webmail	1	(1) Computer	Design
ParAdmin	1	(1) Computer	Validation
Volley-ball game marking sheet	1	(1) Tablet	Design
Mastermind	1 or 2	(1) Computer	Design
Genindexe	All employees	(4) Computers	Design

[12] http://www.genindexe.com/.

- the difference between model and tool components,
- the lack of semantic description,
- the lack of pre-defined value components,
- the lack of computable conditions,
- the weaknesses of object consideration in the dynamics of the model,
- the non-accessibility of data and their semantics.

The implementation of K-MADe was performed in two steps. The first one aimed at coding the kernel following the EXPRESS description of the model. All components of the model as in Lucquiaud (2005) are included. In a second stage, the interface was developed. It allows the use of the model concepts and is expanded to facilitate the modelling process (such as the addition of *unknown* as task executor). In spite of these additions, the tool allows the description of K-MAD task models. Therefore, the implemented K-MAD model (in K-MADe) is very close to the theorical defined K-MAD model.

The K-MADe developers have offered a user manual (http://kmade.sourceforge.net/) that explains the semantics of K-MADe components and how to express them. This documentation aims at facilitating the use of the K-MADe functionalities and also at defining the **semantics** of the K-MAD concepts for future users. The use of K-MADe shows that with the user manual, few components are left with unclear semantics. Only the semantics of the task interruptibility characteristic according to the scheduling operator usage stays unclear. Studies are currently performed in order to define the semantics of this feature in the following version of K-MADe. During the K-MADe development, the major concern was to allow users to perform computable task models. **Each component can thus be defined in order to be computable** when it makes sense to do so (several components cannot be computed as for example task names): task number (as IAWB), task executor (as other task models), scheduling characteristics (optionality (as CTTE and TAMOT), priority, interruptibility (as CTTE) characteristics), scheduling operators (as others task models) and objects. Objects reflect the state of the world. As previously indicated, the task execution may be conditioned by this state of the world, according to the state of objects. As these objects are defined by computable values, they can be used to **express conditions** following pre-defined syntax (see Fig. 3 that shows one of the condition edition windows). These conditions are computed during the simulation of task models. Through the computing of conditions, **objects are taken into account in the dynamics of the model**.

The last limitation of the task model use is **the non-accessibility of data and semantics**. Adding the capability to access task model data and semantics increases the interest of such task model tools. For this reason, since the beginning of the K-MADe design, designers have planned to provide functionalities that allow task model data and semantics to be accessed (Lucquiaud, 2005). Full access to these elements is provided in K-MADe, but advanced exploration functionalities need to be implemented in the tool.

4.2.2.2. Lessons learned from case studies. Increasing the power of task models in incorporating computable objects is a real help in task modelling.

The studied task models express activities of several users, several systems and were chosen in order to design or validate different interactive applications. The design of these wide application types highlights the benefits of the computable object use.

The first benefit is common to all case studies. Objects are used in order to **complete the task scheduling**. In task model tools, task scheduling is mainly expressed through scheduling operators (Section 2.2.1.3). However, some task completions cannot be expressed through scheduling operators only. For example, to design the volley-ball marking sheet task model, objects were mandatory to express the end of a game. A volley-ball game ends when one team wins at least 3 sets, no matter what the score of the other team is. Thus, the number of sets is not pre-determined. This condition for the end of a game can be expressed only because K-MAD scheduling operators and computable objects are jointly used.

In addition to this scheduling role, entities can be computed and therefore, **help the validation**. For example, when designing the *ParAdmin* task model, a dysfunction was detected: the data warehouse could be sorted according to three different algorithms. However, the task that chooses the sorting type was not always performed. By using the simulation tool of K-MADe (which dynamically manipulates the objects), we were not able to define the scenario that uses such data when required.

The simulation of task models that exploit the computable entities (and expressions) **provides a complementary representation** of the hierarchical task tree. From the Genindexe case study, this representation of the state-of-the-world is closer to the user representation of the final application. However, the presentation of the state-of-the-world in the simulation tool (label 1 in Fig. 4) is difficult to understand for users who are not the designer.

Finally, using K-MADe to design the different case study task models highlights the **importance of tools that exploit pre-defined entities**. In K-MADe, we mainly use the grammar-checking and the simulation tools. The grammar-checking tool allows a quick detection of grammar errors (as a wrong pre-condition) and to use the simulation tool to perform scenarios using state-of-the-world entities (as to perform scenarios for each actor). Nevertheless, features that K-MAD offers do not seem to be complete enough to take all situations into account. The case studies highlighted two other **needs concerning users** and **events**. Using K-MADe, users can be associated to a task that specifies who performs it. This type of definition is not adapted to *express activities of a group of users* (i.e. as the *Genindexe Company*). Euterpe allows this definition as a particular agent. K-MADe should also have such kind of multi-user characterization.

Moreover, while events can trigger or suspend the task executions, they cannot totally cancel them whereas some events trigger the *cancelling of tasks* (for example, rainy weather may cause the cancelling of a walk). Only CTTE clearly allows the expression of task cancelling by using the deactivation operator. But, what are the consequences of the use of this action on the state of the world? Does cancelling tasks imply the state of the world goes back to a previous state? Or does it stay in its current state? CTTE does not take into account the state of the world when this operator is used and does not provide answers to these questions. In contrast, prior to allowing task cancelling, K-MADe requires to identify the action to be made on state-of-the-world objects.

4.2.2.3. K-MAD compensates for limitations of other task models. Task models can be used to design interactive applications. To achieve this goal, Sinning et al. Sinning et al. (2007) identified some needs that are common to all type of interactive applications: the **unsuccessful termination of tasks**; the **expression of non-deterministic choices** (decision making without any participation of the user); the **concurrent execution of several "instances" of tasks**. These points cannot be expressed using any task model tool but they are mandatory to design interactive applications (Sinning et al., 2007). This observation leads the authors to suggest add-ons to be developed in CTT. We used K-MAD to perform the task analysis of the case studies in order to show how this model can address these needs. These needs are confronted here to K-MADe solutions.

4.2.2.3.1. Unsuccessful termination of tasks. In the context of K-MADe use, the unsuccessful termination of tasks cannot be specified (users do not want the unsuccessful termination of their tasks and K-MADe was developed to express the achievements of user's

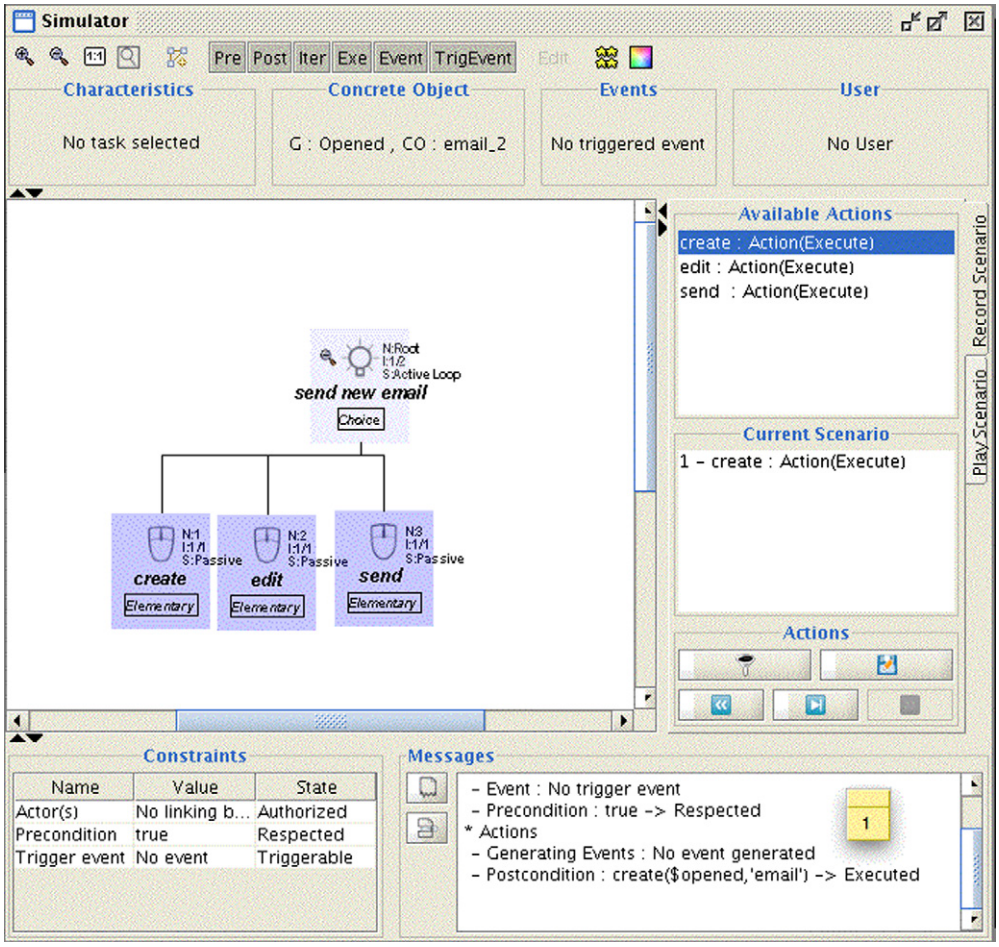

Fig. 4. The K-MADe simulation tool.

goals). As we establish in a previous article (Caffiau et al., 2007), task models are not really suited for expressing the actual dialogue model of interactive applications. While simulation looks like dialogue of applications, many ergonomic considerations impose to largely enrich the dialogue model of application compared to task models (Palanque et al., 2003).

The last two limits (the non-deterministic choices and the execution of several "instances" of tasks) can be expressed using the K-MAD components.

Non-deterministic choices are choices that are made without any participation of users. For example, the feedback for sending an email may be: the indication of the email as a sent email or the message of a sending system error (no connection). K-MADe allows the expressing of this kind of choice: the system task *feedback* is composed of two system sub-tasks with pre-conditions (Fig. 5). As conditions are computed to simulate the task model, the performed system task is the one triggered by the state-of-the-world, independently of the user.

The **instance iteration** is defined as the execution of several "instances" of a task. For example, for the concurrent execution of sending new email (creating, editing and sending emails), the instance iteration corresponds to the concurrent execution of a task several times (iteration characteristics). Each execution manipulates several concrete objects of the same abstract object (in this example, several emails). In our case study, by using the conditions and the *choice* operator, the iterative task *send new email*, allows the creating, editing and sending of several emails at the same time (Fig. 6).

Even though this modelling produces scenarios that create, edit and send several instances of emails (Fig. 7), these instances are visually presented on the task model. Sinning et al. Sinning et al. (2007) propose to add new scheduling characteristic that specifies this particular iteration. Thus, two different kinds of iterations will specify the task iteration characteristics: first, the sequential task iteration (task executions are sequentially performed) as the task model tools currently propose and secondly, the instance task iteration (iteration of instances). This proposition allows the distinction of the two different iteration concepts.

Moreover, while CTTE and TAMOT specify the task iteration as a boolean (without any other information on the iteration) and as integers (minimum and maximum of iterations), K-MADe has two means of defining the task iteration. When the iteration numbers is known and frozen (for example, the task of placing a peg in the current combination in the mastermind game), this number is specified (Reffye et al., 1988 for the *place token* task iteration in Fig. 8) and if this number is not known, the iteration condition is expressed as a mathematical predicate (*While(not(getValue($EvaluatedCombination, $NombrePlaced)==4)) and (card($EvaluatedCombination)<8)*) for the *propose combination* task in Fig. 8.

5. Empirical evaluation of K-MADe usage in designing applications

This section presents a study that evaluates the implications of an approach that integrates computable entities for modelling user activities. This study is exploratory in the sense that it is an overall

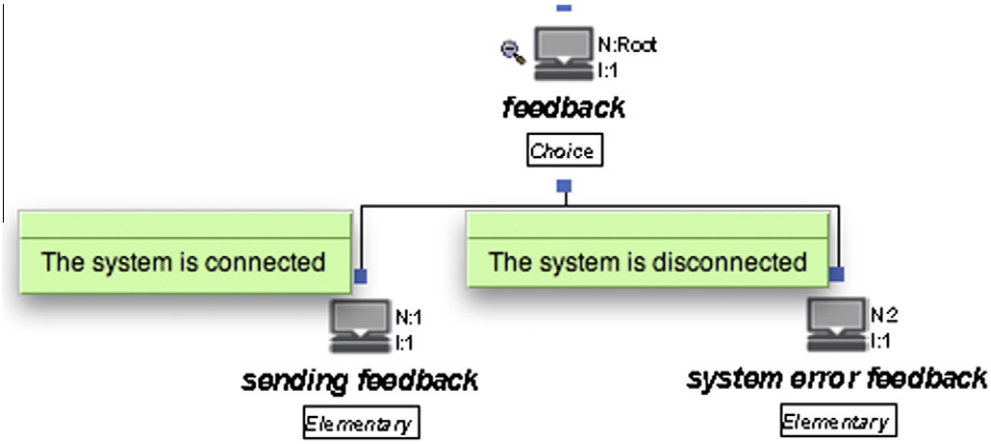

Fig. 5. A "non-deterministic choice" in K-MAD.

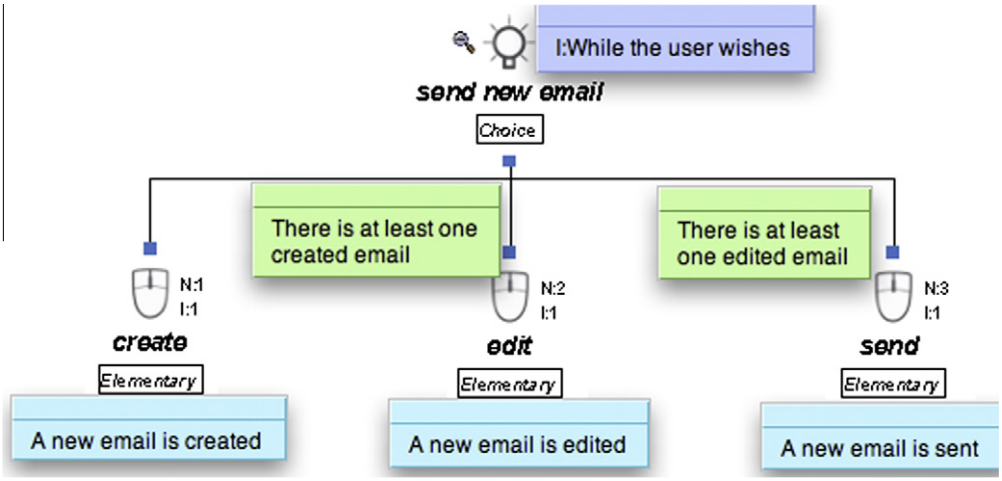

Fig. 6. Iteration of a task manipulating several emails.

examination of potential difficulties encountered during training and usage of a particular task model and tool, without preconceived hypotheses or selection of particular experimental variables. Besides, it is, to our knowledge, the first study of this kind reporting data on actual use of task modelling with a tool. While K-MADe is dedicated to users with different skills (computer scientists, ergonomics experts...), the subjects of our study belong to only one category: HCI students. The target of our evaluation is the use of task models for the design of applications. In order to better understand why computable entities are/can be used, we focus on two aspects: their learning and their usage in the task modelling process. We thus divided our study into two evaluations.

The first evaluation aimed at defining the learning process of the computable aspects during task modelling. In the second evaluation, usage of computable entities in task modelling design was investigated.

5.1. Subjects

Two groups of students took part in this evaluation; their characteristics are summarized in Table 15. All participants were students in their fourth year at university. The first group was composed of 48 bio-informatics students (31 males and 17 females), and the second one of 20 computer science students (that had been studying computer science since their first year at university) (19 males and 1 female). Only one computer science student (participant in the second group) was not a native French speaker. Nevertheless, they all attended the same HCI course.

5.2. The study schedule

The study was divided into two steps. During the training (1 month[13]) on the modelling of task activities using the K-MADe tool, a first evaluation was conducted; once students were trained, a second evaluation was performed to identify their usage of the model. The first step was performed during the lecture and two practical sessions. Four task models were designed during this step (TM1, TM2, TM3 and TM4). The second step was an evaluation of usage of K-MADe computable data (only the TM3 task model).

However the second group (computer science students) only completed the second (and not the first) evaluation. Then, the data of the evaluation of the use of computable entities in the task modelling design were gathered from all students. Fig. 9 summarizes the different steps for each group. We can divide the study into three different categories: the lecture, the practicing of the HCI course and the evaluation session.

[13] It must be noted that this amount of time does not refer to full time training. Only a few hours were spent on this training by each student.

Task treatment: edit
* Executing constraints
- User: Authorized
- Event: No trigger event
- Precondition: (card($opened)>0) -> Respected
*Actions
- Generating Events: No event generated
- Postcondition: (set[email]($opened,($subject:=Hello));
set[email]($opened,($adressee:=lab@lisi.ensma.fr)))->Executed

Task treatment: create
* Executing constraints
- User: Authorized
- Event: No trigger event
- Precondition: true -> Respected
*Actions
- Generating Events: No event generated
- Postcondition: create($opened,'email') ->Executed

Task treatment: edit
* Executing constraints
- User: Authorized
- Event: No trigger event
- Precondition: (card($opened)>0) -> Respected
*Actions
- Generating Events: No event generated
- Postcondition: (set[email_1]($opened,($subject:=Email Sending));
set[email_1]($opened,($adressee:admin@lisi.ensma.fr)))->Executed

Task treatment: send
* Executing constraints
- User: Authorized
- Event: No trigger event
- Precondition: (card($opened)>0) -> Respected
*Actions
- Generating Events: No event generated
- Postcondition: replace[email]($opened,$send) ->Executed

Task treatment: send
* Executing constraints
- User: Authorized
- Event: No trigger event
- Precondition: (card($opened)>0) -> Respected
*Actions
- Generating Events: No event generated
- Postcondition: replace[email_1]($opened,$send) ->Executed

Current Scenario
1-edit: Action(Execute)
2-create: Action(Execute)
3-edit: Action(Execute)
4-send: Action(Execute)
5-send: Action(Execute)

Fig. 7. Example scenario (with dynamics of objects) from task model presented in Fig. 6.

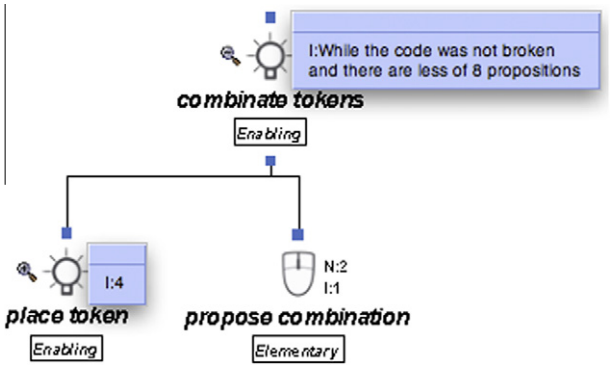

Fig. 8. Specification of the iteration number.

Table 15
characteristics of the two groups of participants.

	Group 1	Group 2
Number	48 (31 males/17 females)	20 (one non-French speaker) (19 males/1 female)
Study level	4th year of French university	4th year of French university
Study domain	Bio-informatics	Computer science

5.2.1. Lecture

This course focuses on user-centered design, with task modelling as part of it. The lecture in task modelling did not aim at teaching the numerous task models; it only focused on the K-MAD

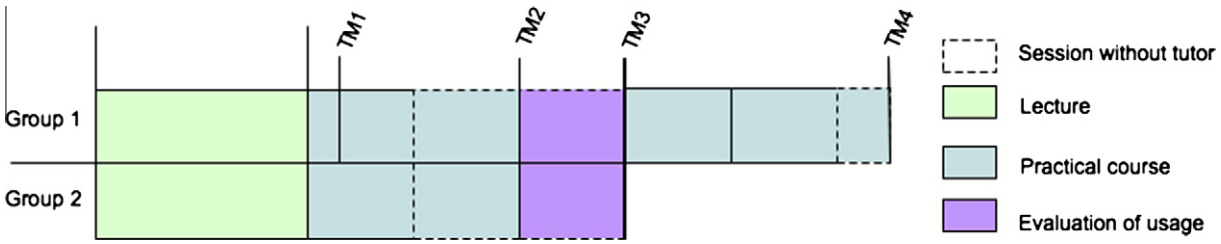

Fig. 9. Steps and the designed task models.

model. This model was explained in details in the lecture (nearly 4 h) and students practiced task modelling using K-MADe during the practical courses. Even if they were not modelling experts, they were more extensively trained on using a task model notation than some ergonomics experts (Couix, 2007).

The second part of this course focused on evaluation (basic concepts of evaluation and main methods used (Nielsen et al., 1993). As the students played the role of observers, the protocol applied in this survey was used as an example in order to facilitate their future work of evaluation. However, as this study was their first practical evaluation, their participation was limited to observation and observation notation. The subjects' notes were completed by other data (user-log and task models).

5.2.2. Training course

During the first practical session, students had to design two different task models. The first one presented their activity before arriving at university each morning (TM1). Few concepts were used to design this task model. They had to decompose this activity without defining objects or conditions. From the design of the second task model (TM2), all K-MADe concepts were used. This second task model described the activity of a clerk in a car rental office (described by textual report of a rental secretary interview). To design this task model, the students were paired and had no time limit, after the end of the practical session (we estimated this time nearly to two hours).

5.2.3. Usage sessions

During the usage evaluation session, students were requested to model the activity for completing a volley-ball game marking sheet (TM3). Instructions for this activity were given at the beginning of the sessions. They were composed of the official instructions of the French Volley Ball Federation (FFVB) and two examples of marking sheets (completed and non-completed ones). Finally, to design the last task model (TM4), students were able to ask questions to the tutors. From this task model, they had to design the software program for a laboratory. Only the first group of students (bio-informatics students) worked on task model TM4.

5.3. Evaluation method

In order to perform this assessment, we used a classical evaluation technique (Nielsen et al., 1993): real-time expert observation of subjects using the tool. In order to evaluate the progression of the use of computable concepts in the apprenticeship of task modelling, a two level observation was conducted: an observation of the first whole group (global observation) and an observation of the behaviour of each student (individual observation). The individual observation of all students (of the first and second groups) allowed the evaluation of the usage of the K-MADe computable components.

Table 16
Observation sheet example.

Type	Observation	Time
FU	The main window is not accessible ("simulation" is noted but the simulation window is not accessible either).	
	=> re-launch K-MADe	14h32
G	Looking for the object definition	14h34
		14h37
F	User does not understand the semantic of the button with shell-hole	14h40

5.3.1. Global observation

After each session of the first group, the tutors noted what they had observed. The global understanding difficulties (wordings and/or concepts) were written in detail (for example, when students did not use iteration condition as a Boolean condition). The tutors also noted the student answers. These notes aimed at helping in the interpretation of other gathered data (as task models).

5.3.2. Individual observation

This observation was set up in the evaluation session (Fig. 9). All students were paired. During the first half part of the session, one student acted as a task model designer (using K-MADe, so named *user*), while the second acted as the evaluation expert (named *observer*). They reversed roles during the second part. Each session lasted 1 hour and a half with a 15 min break between sessions. The modelled activity was the same for all students and was introduced in French at the beginning of the sessions. The case study was the volley-ball game marking sheet. K-MADe was used to model the tasks to be performed.

5.3.2.1. Observers' notes. During the modelling, the observer ensured that the user verbally described his/her modelling process[14] (asking questions (why do you do this? What do you want to do? What are you looking for?) that were given by evaluation instructor), and noted what s/he observed concerning the use of the tool by his/her user (hesitations, exploration in several parts of the software without actions, etc.). In order to help observers in their evaluation, observation sheets were given to them (illustrated in Table 16). These sheets were mainly composed of a three column table corresponding to the three types of information recorded for each observation:

- The type of the observation among a set of defined categories (user goal (G), tool functionalities (F), functionality utilization (FU) and information (I)).
- The observation in textual form.
- The time of observation (reporting the time of the user computer in order to correspond to the user-log (generated with computer hour)).

[14] It should be noted that this method may disturb the user, and change the task.

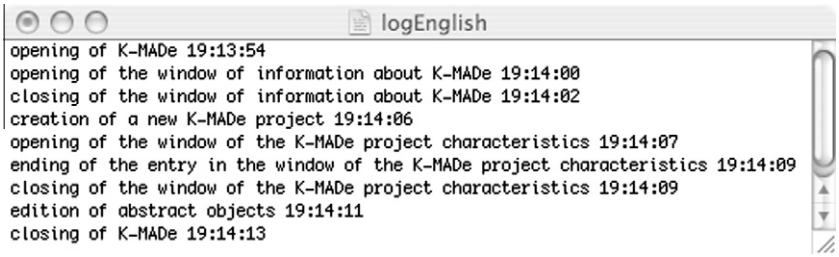

Fig. 10. A user-log produced using K-MADe.

Table 17
The collected data.

Session	Task model	Group1	Group 2
Session 1	TM1	– Tutor notes	– Tutor notes
	TM2	– Tutor notes – Task models	
Session 2	TM3	– User-logs – Observer-student notes – Student exploitation document – Task models – Tutor notes	– Observer-student notes – Student exploitation document – Task models – Tutor notes
Session 3	TM4	Tutor notes tutor notes	–

Table 18
The selection of the task model data.

	Gathered	Used
TM2	11	8
First group TM3 folders	48	46
Second group TM3 folders	20	19

As this study was the first practical evaluation of students, their participation was limited to observation and observation notation. In order to complete the observers' notes and the designed task model, two other types of data were used: user-logs and questionnaires.

5.3.2.2. User-logs. To complete the observations carried out during the evaluation, the users in the first evaluation used a version of K-MADe that produced user-logs. This version allowed the keeping track of user's actions using timestamps and produced a text-file (the user-log). Particularly, this log indicates when the user enters and exits each K-MADe space (task space, abstract objects, edition condition windows (pre, post and iteration)...). Fig. 10 shows an example of this file.

5.3.2.3. Questionnaires. At the end of the evaluation session, the participants of the second group were asked to complete a questionnaire in French about the use of objects. This questionnaire was composed of five questions on definitions, object deletions and conditions. Questions focused on object understanding (When do you use them? Why? And how?).

5.4. Data

Since each session tried to reach different goals, we did not collect the same data for each experiment. In this section, we present the results of each study. Table 17 summarizes the data gathered according to the session and the evaluation goal. The data that could not be used to perform an evaluation was not included in the analysis (Table 18). In the second part of this section, the selection of the data is justified.

5.4.1. Collected data

The goal of the first task model exercise (TM1) is to introduce the use of K-MADe, therefore, we did not collect any specific data from this modelling session. While the second task model (TM2) was designed, students used all concepts of K-MAD for the first time. The task models to analyze the students' understanding of computable concepts of K-MAD were collected.

The third task model (TM3) aimed at reaching two different goals: the knowledge of the task modelling process; the knowledge of the computable entity usage and difficulties. And lastly, in order to evaluate the use of computable entities after the task model apprenticeship, students designed their last task model (TM4).

5.4.1.1. Modelling process. The first group evaluation session (designing TM3 for group 1) aimed at analyzing the task modelling process, particularly when the K-MADe entities are defined and when they are manipulated. In order to obtain this information, user-logs and notes from the observers were used. These two types of information allow the collection of two complementary data. While the observer is focused on usage, what the user goals are and how s/he models conceptually, the user-logs give information on how the K-MADe components are used. Using timestamps on both data, we can determine how users actually use K-MADe tool components.

Moreover, students were requested to exploit their notes and user-logs to write an evaluation report. This report included the modelling process of the observed user, his/her use and usage of the tool, and an analysis of the resulting model. Even though the produced documents were readable and quite organized models, observer notes and user-logs were also collected for expert analysis.

This first group evaluation session helped to determine when K-MADe entities were used in the global modelling process. We did not collect any precise information about their usage. The objective of the second group session was to answer this question.

5.4.1.2. Computable entity usage and difficulties. As for the first group, the user behaviour of the second group subjects was reported in the observer notes and a document was written to report their observations. However, as user-logs do not provide information about entity usage, we did not use them for this session. In order to analyze K-MADe entity usage, we considered two types of data: the models and the questionnaires. The verification of entities in the resulting models indicates the degree of understanding of the object concept. The questionnaire analysis (associated with the student report analysis) keeps records on the difficulties and needs in using objects.

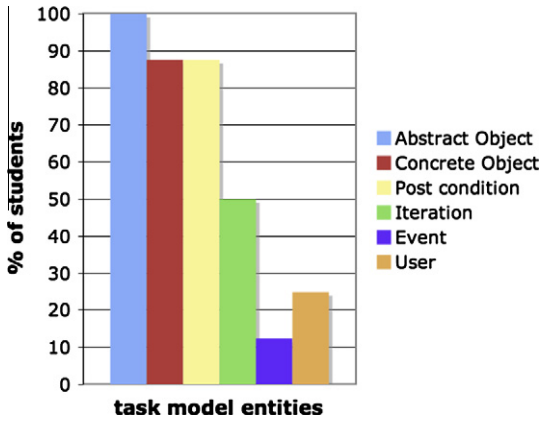

Fig. 11. Percentage of students defining computable entities during the first session.

5.4.2. Selection of data

As students were paired to perform the second task model (TM2), there were 11 models from the first group. However, two task models were not returned. In addition, one returned task model was not completed due to a computer virus so we did not take it into account. Eight TM2 were analyzed.

During the second sessions we collected one folder per user. It included the observer notes, the observer exploitation document, the task model, the user-log (for group 1) and the questionnaire (for group 2). The first evaluation session aimed at gaining some understanding on the modelling process. The data used to infer the user modelling process was mainly taken from user-logs. This file was automatically generated without any technical problem. However, we did not wish to use these user-logs without taking into account the context (reproduced in the observer-student notes and the exploitation documents). Two of the folders were not complete (observer notes or task model were not provided) and therefore were not included in the analysis. We ended up considering 46 out of the 48 folders in our analysis (corresponding to the 48 students of group 1).

The data used for the second group evaluation included all the information in the folders; we could therefore only consider the fully-completed folders. For the first analysis, the observer notes, the student exploitation documents and the task models were only used to help us to give a context to the user-log data but for the second one they were essential. During this evaluation process, we observed that the only non-French speaking student was not able to understand all the directives (this observation was confirmed when he had to complete the questionnaire). He was therefore not considered in the analysis. The second part of the evaluation is based on 19 complete folders.

5.5. Learning of the use of computable aspects

The data collected during all learning sessions allow us to study the computable aspect of the learning time: the use of decomposition operators, entities (objects, events and users) and conditions.

Decomposition operators schedule sub-tasks. The use of the grammar verification tool highlights the users' misconception about this concept. They sometimes mixed up the notion of decomposition and inheritance and thus defined tasks composed of only one subtask. This misunderstanding was explained and corrected by the tutors as soon as it was detected during the first session.

During this first session, in order to design their second task model (TM2) students had to define abstract objects, concrete objects, events, users and to use them to write pre, post and iteration conditions. All these concepts were presented during the lecture and the tutors illustrated the editing of a pre-condition. They also corrected the students' mistakes. We did not evaluate whether students correctly used the entities but only whether they used them or not. Fig. 11 presents the percentage of students who defined computable entities to design the second task model (TM2). As shown in this figure, the **abstract** and **concrete objects** and their use to define *post-conditions* seem to be well understood. The definition and the use of objects require the manipulation of string (to name), set types (to group objects) and basic computer types (to define attributes). All students have a computer science background: the use of these types does not cause any problem. Whilst the necessity of defining entities is understood by students, their roles and links seem to be less assimilated. Concrete objects are used to define conditions; however, 12.5% of students who defined abstract objects did not instantiate them and 12.5% did not use objects to express any conditions. Twenty-five percentage of students thus defined abstract objects without any link to another concept.

Concerning the computable **conditions**, 87.5% of students defined at least one post-condition from the first session onwards. We do not take into account the definition of pre-conditions because tutors used them to illustrate the use of the K-MADe calculator (see Fig. 3). When students did not succeed in defining their conditions using the K-MADe calculator (by using the mathematical syntax), they did it literally using textual definition (Fig. 12). Even though the definition of the K-MAD computable entities and conditions using K-MADe caused some difficulties, the necessity of their use appeared natural for students in order to complete

Table 19
Distribution of students by process.

Did not use any formal entities	Schema 1	Schema 2	Schema 3
12	15	10	9

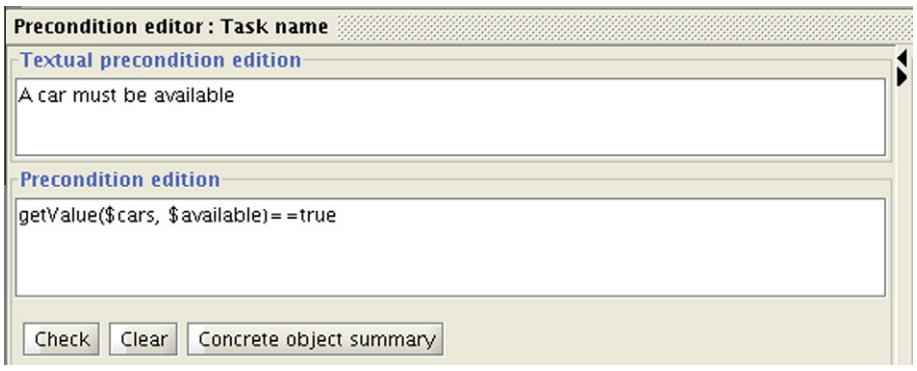

Fig. 12. Edition of a pre-condition.

Schema 1	Schema 2

task tree composition;
while (time is not finished){
edit objects;
define properties using them;
}
	task tree composition;
	edit objects;
	define properties using them;

Fig. 13. Schemas of student task modelling process.

Table 20
Users defining each of K-MADe elements.

	Event	User	Abstract object	Concrete object	Group	Pre	Post	Iteration
Group 1	9	9	34	28	34	18	18	9
Group 2	5	2	19	19	19	16	17	16

the task scheduling (Caffiau et al., 2008). Students of this study had some computer science knowledge. Since this first usage evaluation, we have observed students with different skills. For non-computer science students, while object definition is natural, the use of conditions is not.

Fig. 11 shows that few students used *iteration* conditions, *events* and *users*. Iterations are defined as pre and post-conditions (using the calculators), without more technical difficulties. Events and users are defined with string. The lack of use of these concepts may be due to the teaching. Whilst objects, pre- and post-conditions had been presented (and illustrated) by tutors, iteration, users and events had only been presented in the lecture (without any example).

5.6. Modelling process

After students had discovered the K-MADe tool and the K-MAD model, their task modelling processes were studied during the evaluation session and especially the intervention of computable entities in this process. Prior to identifying the intervention of objects in the task modelling process, it was observed that some students did not define objects. Indeed, 26% of users (12/46) of the first group in the second session did not try to define (or use) any K-MADe computable entities. However, we cannot precisely identify why. Two reasons may explain the absence of these elements in the task model process: the limited duration of the experiment, or the non-assimilation of object concepts. Student notes and reports did not allow us to identify the main reason. Six participants indicated that the sessions were not long enough but others (6/12) did not give any relevant information on the subject.

From the 34 remaining folders, we identified three main schemas followed by users to design task models. Table 19 presents the distribution of students for each process. The most used (44.10% of schemas integrating computable entities) are divided into two steps. Firstly, the user composes the task tree (decomposes tasks). Secondly, s/he iteratively writes entities and associates them with tasks. The steps of the second most used schema (29.4%) are sequential. The user performs the task decomposition prior to defining all entities, and then associates them with the tasks (using conditions). Moreover, an incomplete process (followed by 22% of user-students) is composed of the first two steps of the second schema. The last schema is the iteration of the second one. Fig. 13 summarizes the first and the second schemas.

These observations give us some understanding on the role that objects have in the task modelling process. As an example, the concurrent definition of objects and task tree composition indicates that users associate objects and tasks. Conversely, when the definition and the use of objects are separated from the task tree composition, we can deduce that the user defines objects only to use them for condition expressions. Therefore, objects associate properties and tasks for some users.

5.7. Definition of computable entities (after learning)

The second investigated point is the definition of computable entities to model tasks. We especially aimed at identifying what computable entities are used and why.

5.7.1. What computable entities are used?

From the data gathered in both evaluation sessions (of group 1 and 2), Table 20 shows how many students define each task model component. According to these results, few users integrated the concepts of *events* (20% in the first group and 26% in the second) and *users* (20% and 10.5%) in the task modelling process. On the opposite, the major part of the first group users and all user-students of the second group defined the objects (abstract and concrete objects) in order to model the activity.

To define event and user entities, only one-level strings are used (non-composed definition). Associating them with tasks is easy when using the selection among the defined elements. In contrast, the abstract and concrete objects are composed of several concepts, and their definition is quite hard. The lack of use of events and users cannot be related to their complexity. No explanation can be inferred in that case.

5.7.2. Why are computable entities used?

However, Table 20 also indicates the proportion of users using pre, post and iteration conditions. In the evaluation session, these three types of object manipulation were widely used in the task modelling process (for the second group, 84% defined at least one pre-condition, 89.5% defined at least one post-condition and 84% defined at least one iteration condition). Prior to editing these conditions, the user needs to define the objects (abstract objects, concrete objects and groups). The objects may be defined only to allow the definition of conditions. Therefore, the users did not see objects as a part of tasks but as a way of defining conditions.

From these figures, it can be observed that for the majority of students, it is natural to define objects in order to complete the semantics of scheduling operators. As an example, when they designed the task model of the volley-ball marking sheet, all students defined conditions using objects and conditions to express the end of a match.

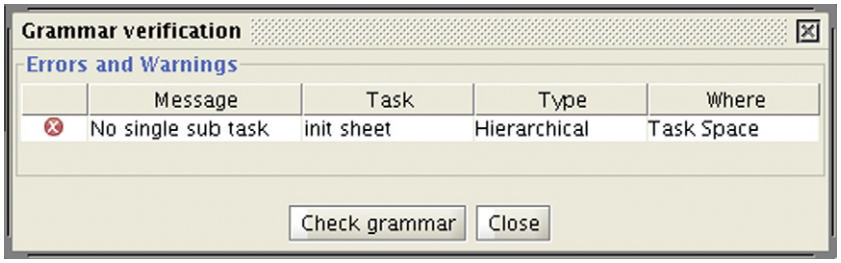

Fig. 14. View of grammar inconsistency detection.

The study of the proportion of definition of K-MADe concepts in the two sessions shows a difference between the two groups. In the first evaluation group, 26% of users did not use any entity. Whilst all participants followed the same lecture, the discrepancy cannot be explained solely by the teaching. However, these groups do not have the same background. Figures shown in Table 20 clearly indicate that computable object definitions are easier to be used by pure computer science students.

5.8. The use of computable aspects to validate task models

The computable aspects in task modelling allow verifications on task models in order to detect discrepancies, for example, the fact that a task is composed of only one subtask (in grammar verification tool, Fig. 14). K-MADe offers tools that perform this type of verification prior to allowing the access to the simulation tool.

The last observations of this evaluation concern the use of the grammar verification tool and the simulation tool. We present why and when students used these two K-MADe tools in the task model processes.

5.8.1. Why are correction tools used?

During the task modelling training, these tools may be used so that the model complies with the model syntax.

When this step was completed (from TM3), 62.5% of student continued to use at least one correction tool, and 12.5% indicated that they did not because they did not have enough time. The coherence verification tool detects one error in 90% of the cases which indicates that users need this type of tools to design their models. Moreover this coherence verification tool is the most used correction tool (87.5% of students stopped the simulation tool as soon as any coherence error was detected).

5.8.2. When are correction tools used?

The study of user-logs shows that users triggered correction tools, especially after they decomposed their tasks (between lines 1 and 2 in the schemas of Fig. 13) and when they considered the task modelling completed. These uses correspond to the errors detected by the coherence verification tool.

Moreover, the computable conditions were systematically checked (sometimes many times) when they were written in order to verify their syntax.

6. Summary of results and research avenues

In this section, we first summarize the lessons learned from the systematic and empirical evaluation of K-MAD/K-MADe. Following these evaluations, several improvements can be planned. They particularly concern the definition and use of the computable aspects and the use of K-MAD task models.

6.1. Summary of results

The use of K-MADe to design task models of interactive applications (for our case studies and the assessments) provided results that correspond to two main points: results about the definition of computable entities and results about their use.

K-MADe computable entities are mainly defined after the task decomposition, to **add complementary information**. We observed that the computable entities are defined in order to add semantics to the K-MADe tasks. **Task definition is not disjoined from the definition of entities that are manipulated to perform the tasks**. For example, from the interviews (Sebillote, 1995) performed to design the *Genindexe* task model, the tasks are described with their handled objects ("Then, I send the invoice"). However, the definition of tasks without any object limits the understanding semantics to the task name and is thus dependent on the task model reading. On the contrary, the association of objects (used in the conditions) precisely describes the task behaviour (execution conditions and effects). Therefore, the adding of objects to the tasks improves the understanding and consequently the communication between all the application design actors (one of the six evaluation points in Balbo et al. (2004)).

In addition to this semantic goal, objects (and so conditions) are also used in order to complete the schedule. This contribution rapidly appears to task model designers during our case studies (this role of objects is mandatory in 4/5 of case study task models) as the student usage (all students naturally used computable entities to define the end of a volley-ball game). Thus, the analysis of the presented evaluations shows that for designers, the **K-MADe features are mandatory to design task models**.

Finally, the main goal in the use of computable entities is the capability to compute them in order to **perform verification**. As the empirical evaluation showed, the most used entities are the ones that can be computed and used in the K-MADe verification tools.

K-MADe offers two tools for the design of task models in order to support interactive application design or validation: the grammar-checking tool and the simulation tool. During the evaluation study, the grammar-checking tool was widely used (only one student performed a scenario from the simulation tool). The grammar-checking tool checks that the designed model complies with the model syntax. This function acts for two roles. First, during the training, it **helps the students to use the task model syntax correctly**. And secondly, during all the other task model designs, it quickly and **automatically detects the syntactic designer mistakes** (the grammar-checking tool is mainly triggered at the end of the task modelling steps).

Even if students did not use the simulation tool, it was an important tool when, some scenarios needed to **be validated by the application domain experts** (for the *Genindexe* and the *ParAdmin* task model). These experts did not have any HCI skills and did not have any knowledge of task models. In order to explain the designed task models and to verify that it was designed according to the gathered activities, we used the K-MADe simulation tool and we generated task execution traces that represent the user activities. Producing scenarios manually cannot be considered as full test

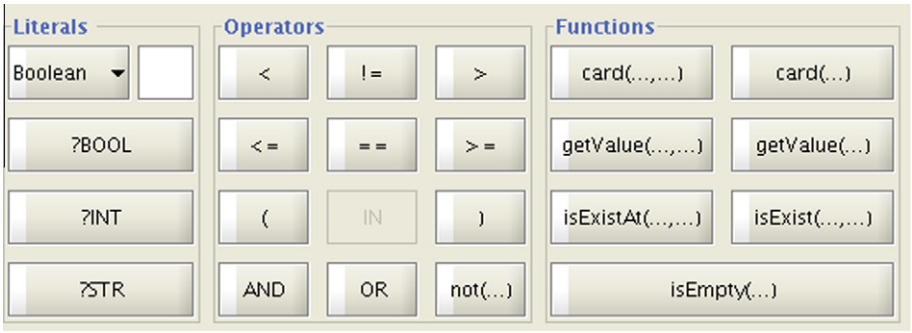

Fig. 15. Calculator of the condition editor.

coverage. But the simulator could be used to verify that the scenarios described in the requirements are correctly implemented in the task model. Nevertheless, using scenarios with the K-MADe simulator proved to be usable with domain experts at a validation stage. During simulation of the designed task models, domain experts verbally expressed that they were confused. Task model designers had to to explain the task model simulation. A specific assessment should be performed in order to evaluate the domain expert use of the K-MADe evaluation tool from this first observation.

6.2. The K-MAD computable aspects

The usage of the K-MADe tool highlighted the need for completing the K-MAD model in order to take all concepts required to model any activity into account. For example, *Users* are integrated in the K-MAD model and they can be linked to the tasks specifying the task actors. This point of view corresponds to the description of **one** user activity. However, to design a task model of activities concerning a whole group of users (as in a company) this definition of actors is not appropriate. K-MAD will be modified to include the definition of groups of users as potential actors. Following the wish to associate a tool closer to the task model, K-MADe needs also to be extended using the group-actor concept.

The use of the tool highlights also the lack of semantics of some model entities. For example, using and teaching (Caffiau et al., 2008) task modelling shows the difficulty to understand the event entity and furthermore, to use it. All limits in K-MAD components semantics highlighted in our evaluations must be addressed in the model prior to be implemented in the tool.

The last point concerning the definition of the K-MADe computable aspects and requiring more specific studies concerns the *object* entity. As the evaluation showed, the main difficulties of usage concern the definition and the use of objects. Two main reasons can explain that. First, the concepts used to define objects are not always clear and necessary from the user point of view. For example, *concrete objects* are created (and manipulated) only through a *group* of *abstract objects*. However, for the user, the definition of an only-one-element group does not have any meaning. Secondly, these objects are used to define conditions and some users do not define objects when the objects are not used to define any condition. However, the definition of conditions requires the use of calculators and their use is non-intuitive (see Fig. 15) for users, which causes difficulties, errors and non-use of computable conditions. The definition and use of objects will be the topics of subsequent work.

6.3. The use of K-MAD task model

The K-MADe tool has to be very close to the task model and has to help the design and the use of its concepts. In order to improve this second point, we selected three groups of tools according to three kinds of needs: the needs for the design step; the needs for the task model validation by the domain expert; and the needs for the K-MADe task model usage.

6.3.1. Needs for the design step

In order to help during the design stage, two tools might be added to K-MADe. These tools concern the validation of the task model semantic expression and the querying of the task models.

The K-MAD syntax has to be followed to design task models using K-MADe. Designers have to observe this semantic expression to benefit from the validation tools. A tool able to automatically detect the semantic errors is planned. This tool will check the observation of the constraint coherence (such as a subtask pre-condition that is incoherent with the parent task pre-condition) and of the scheduling (such as the fact that all tasks are reachable).

In order to validate the design of task models and to evaluate the consequences of modification(s), the querying functionality and the recording of queries should be suitable tools for model validation. These queries allow designers to compare their task models with their objectives. Euterpe allows querying task models such as: where the object O is used; or what the tasks with S sub-tasks are. The results of these queries are displayed but cannot be recorded. Future developments of K-MADe will integrate the query definition and recording.

6.3.2. Needs for domain expert validation

Taking into account state-of-the-world objects in K-MADe allows us to be closer to real-life activities. The closeness between the real-life world and the model is particularly important for the validation of task models by domain expert users. This validation can only be performed using the simulation tool and the scenario recording/replaying. As written above, computable conditions are automatically computed when the K-MADe simulation tool is used. However, other functionalities require to be added in order to help the task model validation. For example, the design of task models requires scheduling tasks. Taking into account the modification of the scheduling operators directly in the simulation tool could allow us to see the consequences of the changes.

Moreover, the use of the K-MADe simulation tool to validate the task models of our case studies by users (in particular by the employees of *Genindexe*) shows that the simulation tool (Fig. 4) is not easy to use for non-I.T. specialists. While the presentation of the task model data (schedule, conditions, objects...) is of interest for designers, the user validation focuses on the performed scenarios. The needs for task model validation by expert domain users will be taken into account by the study of a specific task model view of the simulation tool.

6.3.3. Needs for K-MADe task model usage

The task model design is iterative. These models may be modified and consequently the storage of previous versions is important. Moreover, the management of several versions of the description of activities makes team work easier. Users can thus compare two versions of the description of the same activity. The development of tools that allow these actions to be performed could improve the use of K-MADe thus facilitating its usage and the description of common activities.

7. Conclusion

Ergonomics and software designers perform task analysis during several stages of the software cycle life: analysis, design, development, validation... The activity description is expressed using task models. To represent real-life activities, these models must integrate the concepts of *tasks*, *objects*, *actors* and *events*. Concurrently, some tools associated to these models were developed. Owing to the lack of semantics and the distance[15] between task models and tools, these tools are not often used (Couix, 2007).

In order to address these issues, the K-MAD model and its associated tool K-MADe have been developed. K-MAD includes computable aspects that provide the simulation of the more realistic models that take into account conditions, events and evolutions of the state-of-the-world.

To evaluate the impact of these points, case studies and usage evaluations have been performed. These studies focus on the learning of the computable aspects; the role that they play in task modelling process and validation; and the expressive power of K-MAD.

The data gathered from these studies show the need for computable attributes to model and validate task models (to design their first task models, 100% of users needed to define abstract objects) but also the difficulties to use the tool to define them (the percentage of users who defined abstract objects declined to 74% for the same group when users were not helped by instructors).

The analysis of the data also indicates that the definition and use of computable aspects are easier (using K-MADe) for computer scientists than for others, pointing out one of the usability improvement issues. In that respect, several studies are planned by modifying the interaction techniques and the presentation of information. In addition to the improvement of tool usability, our studies showed the need to better define the semantics of several concepts (for example the events) and to complete the expressive power of the model.

To complement the studies presented in this paper, several aspects of task model use need to be explored. The case studies did not integrate collaborative work. The expressive power of K-MAD requires to be tested to check if it allows this type of activities to be expressed or whether it has to be enhanced (as the CTT notation (Sinning et al., 2007).

Moreover, the participant's skills in computer science do not allow us to generalize our observations to all users: as we have shown before, a minor difference of skills considerably modifies the usage of objects (see Table 20). In order to gain a broader point of view, the same type of evaluations with other background participants should be conducted.

As indicated previously, the use of computable aspects in the expression of activities allows a more realistic description (the adding of conditions completes the scheduling which is expressed by the operators) but their integration in the task model opens other perspectives. Several studies exploiting the increase of expressive power are currently underway in order to check the interactive software dialogue.

This study is exploratory in the sense that it is an overall examination of potential difficulties encountered during training and usage of one particular task model and tool, without preconceived hypotheses or selection of particular experimental variables.

Besides, it is, to our knowledge, the first study of this kind that reports data on actual use of task modelling with a tool. For future work, we plan to consider longitudinal studies, which might bring highlights on task based tools usage during modelling.

And finally, integrating task-oriented design in actual industrial projects may require to develop further links (pathways, models, etc.) between the user task level and the business modelling level, particularly for applications that deal jointly with "front office" and "back office", with different goals and different user types.

[15] There are often some differences between the theoretical model (as published) and the implemented model in tools. These differences create a "distance" between defined models and implemented models.

References

Abed, M., Tabary, D., Kolski, C., 2004. Using formal specification techniques for the modeling of tasks and the generation of human–computer user interface specifications. In: Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human–Computer Interaction. Lawrence Erlbaum Associates, pp. 503–530 (Chapter 25).

AMBOSS, 2008. AMBOSS Electronic Reference. <http://wwwcs.uni-paderborn.de/cs/ag-szwillus/lehre/ws05_06/PG/PGAMBOSS> (last accessed 27.07.08).

Annett, J., 2004. Hierarchical task analysis. In: Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human–Computer Interaction. Lawrence Erlbraum Associates, pp. 67–82 (Chapter 3).

Annett, J., Duncan, K., 1967. Task analysis and training design. Occupational Psychology 41, 211–221.

Balbo, S., Ozkan, N., Paris, C., 2004. Choosing the right task-modeling notation: a taxonomy. In: Diaper, D., Stanton, N.A. (Eds.), The Handbook of Task Analysis for Human–Computer Interaction. Lawrence Erlbaum Associates, pp. 445–466 (Chapter 22).

Baron, M., Lucquiaud, V., Autard, D., Scapin, D., 2006. K-MADe: un environnement pour le noyau du modèle de description de l'activité. In: Proceedings of IHM'06. (Eds.). Montréal, Canada 18–21 avril 2006. ACM Publishers, pp. 287–288.

Bellatreche, L., Boukhalfa, K., Caffiau, S., 2008. ParAdmin: Un Outil d'Assistance à l'Administration et Tuning d'un Entrepôt de Données. In: Proceedings of EDA 2008, Toulouse.

Biere, M., Bomsdorf, B., Szwillus, G., 1999. Specification and simulation of task models with VTMB. In: CHI'99 Extended Abstracts on Human Factors in Computing Systems (Pittsburg, Pennsylvania, May 15–20, 1999). CHI'99. ACM, New York, pp. 1–2.

BPMN 2.0, 2010. BPMN 2.0 Electronic Reference. <http://www.bpmn.org/> (last accessed 25.01.10).

Caffiau, S., 2009. Approche dirigée par les modèles pour la conception et la validation des applications interactives : une démarche basée sur la modélisation des tâches. phD thesis. LISI/ENSMA. Poitiers. pp. 240.

Caffiau, S., Girard, P., Scapin, D., Guittet, L., 2007. Generating interactive applications from task models: a hard challenge. In: Proceedings of TAsk MOdels and DIAgrams (TAMODIA). Toulouse, France Springer Berlin/Heidelberg, pp. 267–272.

Caffiau, S., Girard, P., Scapin, D.L., Guittet, L., Sanou, L., 2008. Assessment of object use for task modeling. In: Proceedings of Engineering Interactive Systems (HCSE 2008 and TAMODIA 2008), Pisa, Italy (LNCS 5247), September 2008. Springer, pp. 14–28.

Caffiau, S., Scapin, D.L., Sanou, L., 2008. Retour d'Expérience en Enseignement de la Modélisation de Tâches. In: Proceedings of ERGO'IA. Biarritz, pp. 135–143.

Calvary, G., Coutaz, J., Thevenin, D., 2001. A unifying reference framework for the development of plastic user interfaces. In: Proceedings of Engineering for Human–Computer Interaction (8th IFIP International Conference, EHCI'01, Toronto, Canada, May 2001. Canada Springer, pp. 173–192.

Carroll, J.A., 2000. Making use: scenario-based design of human–computer interactions. The MIT Press.

Couix, S., 2007. Usages et construction des modèles de tâches dans la pratique de l'ergonomie: une étude exploratoire. Master thesis report (only in French). <http://www.biomedicale.univ-paris5.fr/taskmodelsurvey/accueil/index.php?page=accueil&hl=en_US> (last accessed 29.04.10).

Delouis, I., Pierret, C., 1991. Emad : Manuel de référence Institut National de Recherche en Informatique et en Automatique.

Diaper, D., 2004. Understanding task analysis for human–computer interaction. In: Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human–Computer Interaction. Lawrence Erlbaum Associates Inc., pp. 5–48.

Dittmar, A., Forbrig, P., Heftberger, S., Stary, C., 2005. Support for task modeling – a "Constructive" exploration. In: Bastide Remi, Palanque Philippe A., Roth Jörg (Eds.), Engineering Human Computer Interaction and Interactive Systems, Joint Working Conferences EHCI-DSVIS July 11–13 2004. Hamburg, Germany, pp. 59–76.

Dix, A.J., 1991.Formal Methods for Interactive Systems. Academic Press, p. 384. ISBN: 0122183150.
Dix, A., 2008. Tasks = Data + Action + Context: Automated Task Assistance through Data-Oriented Analysis (invited paper). In: Proceedings of Engineering Interactive Systems 2008 (HCSE 2008 and TAMODIA 2008), Pisa, Italy (LNCS 5247), September 2008. Springer, pp. 1–13.
Gamboa, R.F., 1998. Spécification et implémentation d'ALACIE : Atelier Logiciel d'Aide à la Conception d4interfaces Ergonomiques (Thesis). Paris XI.
Gamboa, R.F., 1998. ALACIE : Manuel d'Utilisation. INRIA.
Gamboa, R.F., Scapin, D.L., 1997. Editing MAD* task description for specifying user interfaces, at both semantic and presentation levels. In: Proceedings of Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'97), 4–6 June. Springer-Verlag, Granada, Spain, pp. 193–208.
Giese, M., Mistrzyk, T., Pfau, A., Szwillus, G., Detten, M., 2008. AMBOSS: a task modeling approach for safety-critical systems. In: Proceedings of Engineering Interactive Systems (HCSE 2008 and TAMODIA 2008), Pisa, Italy (LNCS 5247), September 2008. Springer, pp. 98–109.
Hammouche, H., 1995. De la modélisation des tâches utilisateurs à la spécification conceptuelle d'interfaces Homme-Machine (Thesis). Université de Paris VI.
Heinrich, M., Winkler, M., Steidelmüller, H., Zabelt, M., Behring, A., Neumerkel, R., Strunk, A., 2007. MDA applied: a task-model driven tool chain for multimodal applications. In: Marco Winckler (Eds.), Task Models and Diagrams for User Interface Design: 6th International Workshop, proceedings of TAMODIA 2007, Toulouse, France, November 7–9, 2007. Springer, pp. 15–27.
Hix, D., Hartson, H.R., 1993. Developing User Interfaces: Ensuring Usability through Product & Process. John Wiley & Sons Inc.
IBM Task Modeler, 2010. IBM Task Modeler Electronic Reference. <http://developer.tibco.com/business_studio/default.jsp> (last accessed 25.01.10).
ISO EXPRESS, 1994. The EXPRESS Language Reference Manual.
ISO Systems, 1984. I. I. P. Definition of the Temporal Ordering Specification Language LOTOS.
Jambon, F., Brun, P., Ait-Ameur, Y., 2001. Spécification des systèmes interactifs. In: Hermès-Lavoisier (Eds.), Analyse et Conception de l'IHM (Chapter 6).
John, B.E., Kieras, D.E., 1996a. Using GOMS for user interface design and evaluation: Which technique? ACM Transactions on Computer–Human Interaction, 287–319.
John, B.E., Kieras, D.E., 1996b. The GOMS family of user interface analysis techniques: comparison and contrast. ACM Transactions on Computer–Human Interaction, 320–351.
Johnson, H., Johnson, P., 1993. ADEPT-advanced design environment for prototyping with task models. In: Proceedings of Human Factors in Computing Systems (InterChi'93), Springer, p. 56.
Kieras, D.E., 2004. GOMS models for task analysis. In: Diaper, D., Stanton, N., (Eds.), The Handbook of Task Analysis. pp. 83–116 (Chapter 4).
K-MADe, 2006. K-MADe Electronic Reference. <http://kmade.sourceforge.net/> (last accessed 14.04.09).
Limbourg, Q., Vanderdonckt, J., 2004. Comparing task models for user interface design. In: Diaper, D., Stanton, N.A. (Eds.), The Handbook of Task Analysis for Humain–Computer Interaction. pp. 135–154.
Limbourg, Q., Pribeanu, C., Vanderdonckt, J., 2001. Uniformation of Task Models in Model-Based Approaches. Université catholique de Louvain.
Lucquiaud, V., 2005. Sémantique et Outil pour la Modélisation des Tâches Utilisateur: N-MDA (Thesis). University of Poitiers. p. 285.
Mahfoudhi, A., Abed, M., Tabary, D., 2001. From the Formal Specifications of User Tasks to the Automatic Generation of the HCI Specifications.
Minsky, M., 1988. The Society of Mind. First Touchstone Edition. Simon & Schuster, Inc.
Molina, A.I., Redondo, M.A., Ortega, M., 2005. Analyzing and modeling user task in DomoSim-TPC system for adapting to mobile devices. In: Navarro, R., Lorés, J. (Eds.), HCI Related Papers of Interaction 2004. Springer, pp. 221–241.
Mori, G., Paternò, F., Santoro, C., 2003. Tool support for designing nomadic applications. In: Proceedings of Intelligent User Interfaces (IUI'2003). Miami, Florida 12–15 January 2003, pp. 141–148.
Mori, G., Paternò, F., Santoro, C., 2004. Design and development of multidevice user interfaces through multiplelogical descriptions. IEEE Transactions on Software Engineering, 507–520.
Nielsen, J., 1993. Usability Engineering. Academic Press, Boston. ISBN: 0-12-518405-0.
Norman, D.A., Draper, S.W., 1986. User Centered System Design. Lawrence Erlbraum Associates. ISBN: 0898598729.
Object Management Group, 1997. The Unified Modeling Language.
Palanque, P., Bastide, R., Winckler, M., 2003. Automatic Generation of Interactive Systems: Why A Task Model is not Enough Human–Computer Interaction. Design Approaches, Methods and Tools. Lawrence Erlbraum Associates. pp. 198–202.
Paternò, F., 2001. Model-Based and Evaluation of Interactive Applications. Springer. ISBN: 1-85233-155-0.
Paterno, F., 2004. ConcurTaskTrees: an engineered notation for task models. In: Diaper, D., Stanton, N.A. (Eds.), The Handbook of Task Analysis for Human–Computer Interaction. Lawrence Erlbaum Associates, pp. 483–501 (Chapter 24).
Paternò, F., Faconti, G.P., 1992. On the use of LOTOS to describe graphical interaction. In: Monk, Diaper, Harrison (Eds.), People and Computers VII: Proceedings of the HCI'92 Conference, Cambridge University Press, pp. 155–173.
Paternò, F., Santoro, C., 2002. One model, many interfaces. In: Proceedings of Computer-Aided Design of User Interfaces (CADUI'2002). Valenciennes, France May 15–17. Kluwer Academics, pp. 143–154.
Paternò, F., Mancini, C., Meniconi, S., 1997. ConcurTaskTrees: a diagrammatic notation for specifying task models. In: Proceedings of IFIP TC13 Human–Computer Interaction Conference (INTERACT'97). Sydney, Australia, pp. 362–369.
Reffye, P.D., Edelin, C., Françon, J., Jaeger, M., Puech, C., 1988. Plant models faithful to botanical structure and development. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques. pp. 151–158.
Scapin, D.L., 1988. Vers des outils formels de description des tâches orientés conception d'interfaces. Institut National de Recherche en Informatique et en Automatique.
Scapin, D., Bastien, J.-M.C., 2001. Analyse des tâches et aide ergonomique à la conception: l'approche MAD* (chapitre 3). Analyse et conception de l'I.H.M./Interaction Homme-Machine pour les S.I. vol.1. C. Kolski (Eds.). Hermès Science.
Scapin, D.L., Pierret-Golbreich, C., 1989a. MAD : Une méthode analytique de description des tâches. Proceedings of Colloque sur l'Ingénierie des Interfaces Homme-Machine (IHM'89). Sophia-Antipolis, France, pp. 131–148.
Scapin, D.L., Pierret-Golbreich, C., 1989b. Towards a method for task description: MAD. Working with display units.
Sebillote, S., 1995. Methodology guide to task analysis with the goal of extracting relevant characteristics for human–computer interfaces. International Journal of Human–Computer Interaction 7 (4), 341–363.
Sebillotte, S., 1991. Décrire des tâches selon les objectifs des opérateurs, de l'interview à la formalisation. Le Travail Humain 54 (3), 193–223.
Sebillotte, S., 1994. Méthodologie pratique d'analyse de la tâche en vue d'extraction de caractéristiques pertinentes pour la conception d'interfaces.
Sebillotte, S., Scapin, D.L., 1994. From users' task knowledge to high-level interface specification. International Journal of Human–Computer Interaction 6 (1), 1–15.
Shepherd, A., 1989. Analysis and training in information technology tasks. In: Diaper, D. (Eds.), Task Analysis for Human–Computer Interaction. pp. 15–55.
Shepherd, A., 1995. Task analysis in HCI tasks. In: Monk, A.F., Gilbert, N. (Eds.), Perspectives in HCI. Academic Press.
Shneiderman, B., Plaisant, C., 2009. Designing the User Interface, fifth ed. Pearson. 580.
Sinning, D., Wurdel, M., Forbrig, P., Chalin, P., Khendek, F., 2007. Practical extensions for task models. In: Proceedings of TAMODIA 2007. Toulouse, pp. 42–55.
Tarby, J.C., Barthet, M.F., 2001. Analyse et modélisation des tâches dans la conception des systèmes d'information: la méthode Diane+. Analyse et conception de l'IHM, interaction pour les Systèmes d'Information. HERMES. (Chapter 4).
TaskArchitect, 2010. TaskArchitect Electronic Reference. <http://www.taskarchitect.com/> (last accessed 25.01.10).
Tibco Business Studio, 2010. Tibco Business Studio Electronic Reference. <http://developer.tibco.com/business_studio/default.jsp> (last accessed 25.01.10).
TKS, 2010. TKS Electronic Reference. <http://www.cs.bath.ac.uk/~hci/TKS/publications.html> (last accessed 25.01.10).
Van der Veer, G.C., 1996. GTA: groupware task analysis – modeling complexity. Acta Psychologica, 297–322.