Towards Uniformed Task Models in a Model-Based Approach

Quentin Limbourg[1], Costin Pribeanu[1,2], and Jean Vanderdonckt[1]

[1] Université catholique de Louvain, Institut d'Administration et de Gestion
Place des Doyens, 1 - B-1348 Louvain-la-Neuve, Belgium
limbourg, vanderdonckt@qant.ucl.ac.be
[2] National Institute for Research and Development in Informatics
Bd Averescu 8-10 - R-71316 Bucharest, Romania
pribeanu@acm.org

Abstract. Multiple versions and expressions of task models used in user interface design, specification, and verification of interactive systems have led to an ontological problem of identifying and understanding concepts which are similar or different across models. This variety raises a particular problem in model-based approaches for designing user interfaces as different task models, possibly with different vocabularies, different formalisms, different concepts are exploited: no software tool is able today to accommodate any task models as input for a user-centred design process. DOLPHIN is a software architecture that attempts to solve this problem by introducing uniform task models. A series of representative task models was first selected. The meta-models of these individual task models were then designed and merged into a uniformed task meta-model. Semantic mapping rules between individual task meta-models and the uniformed task meta-model allow DOLPHIN to read and understand any potential task model towards its exploitation in a model-based approach.

1 Introduction

Human-Centred Design (HCD) has yielded to many forms of design practices where information about the context of use is considered. Among these, task analysis is widely recognised as one fundamental way not only to ensure some HCD, but also to improve the understanding of how a user may interact with a user interface (UI) for accomplishing a given interactive task. For this purpose, many task analysis methods have been introduced from disciplines having different backgrounds, different concerns, and different focuses on task [14, 17].

Task analysis typically results in a task model, which is a description of any interactive task to be accomplished by the user of an application through the application's UI. Individual elements in the task model represent specific actions that the user may undertake. Information regarding subtask ordering as well as conditions on task execution is also included in this model. A task model may be issued from [17]:

- *Cognitive psychology*, where it is aimed at improving the understanding of how users may interact with a given UI for carrying out a particular interactive task from a cognitive viewpoint. This model is considered useful for identifying cognitive processes and structures which are manipulated by a user when carrying out a task, for understanding how a user may dynamically change these processes and structures as the task is proceeding, and for predicting cognitive work load.
- *Task allocation*, where it is intended to plan and distribute working tasks in time and space among workers in a particular organisation.
- *Software engineering*, where it captures relevant task information in an operational form which is machine processable.
- *Ethnography*, where it focuses on how human users could behave with a particular UI in a given context of use, also possibly interacting with other users.

Many different task models, task analysis methods, and supporting tools have been introduced as reported in the CHI'98 workshop [3] on task modelling approaches. This situation leads to a series of important shortcomings, among them are:

- Lack of understanding the basic concepts: in each individual task model the rationale behind their method, their entities, their relationships, their vocabularies, the intellectual operations involved for task modelling.
- Heterogeneousness of these concepts as they were initiated by various methods issued from various disciplines.
- Difficulty to match concepts across two individual task models or more: it is even likely that sometimes no matching across these concepts can be established.
- Lack of software interoperability: since task modelling tools do not share a common format, they are only restricted to those task models that are expressed according to their accepted formats. For instance, a task model drawn in a graphical editor cannot be reused in a simulation tool.
- Reduced communication among task analysts: due to the lack of software interoperability, a task analyst may experience some trouble in communicating the results of a personal task analysis to another analyst or stakeholder of the UI development team. In addition, any transition between persons may generate inconsistencies, errors, misunderstandings, or inappropriate modelling.
- Duplication of research and development efforts: where teams conduct research and development on their own task models, it is likely that each research and development effort needs to be reproduced for their specific task models, instead of benefiting from each other. This shortcoming is particularly important for software development efforts which are resource-consuming.

To address the above shortcomings, three major goals were set:

1. To provide an improved conceptual and methodological understanding of each individual task model and their related concepts.
2. To establish semantic mappings between different individual task models so as to create a transversal understanding of their underlying concepts independently of their peculiarities. This goal involves many activities such as vocabulary translation, expressiveness analysis, identification of degree of details, identification of concepts, emergence of transversal concepts, and task structuring identification.
3. To rely on these semantic mappings to develop a User Interface Design Assistant (UIDA) which accommodates any type of individual task model as input. This UIDA should help designers and developers to derive presentation and dialog from any starting task model, whatever its model type. The ultimate goal of this UIDA is to capitalise design knowledge valid for each task model into a single tool and to avoid reproducing identical development effort for each individual task model.

The remainder of this paper is structured as follows: Section 2 summarises open questions regarding task modelling and reports on some previous work done regarding the question of understanding task models. Section 3 defines the differences between an individual task model and a uniformed task model and presents the hypotheses, the method followed towards uniforming task models. Section 4 applies this method to selected individual task models to come up with their corresponding meta-model. Section 5 sums up the results provided by the meta-models of individual task models into a uniformed task meta-model, along with semantic mappings between corresponding concepts. Section 6 outlines how the final meta-model and the semantic mappings are exploited in DOLPHIN, a UIDA that accommodates any individual task model as input.

2 Related Work in Task Modelling

In general, a task is understood as an activity performed by people to accomplish a certain goal. A task could be further decomposed resulting in smaller tasks corresponding to lower level goals. Task decomposition is usually represented as a tree. Inner tasks are said to be *composite* (or abstract, complex) while the leaves are *elementary tasks*, which in turn are decomposed in actions performed upon objects.

As pointed out by activity theory [13], tasks are goal driven, being performed consciously while actions depend on operational conditions of the task and become automatic by practice. However, this distinction is rarely used in the practice of task analysis for UI. In this respect, most task models do not make a clear distinction between task and actions but rather they decompose tasks up to the level that is relevant for the analysis. Goal hierarchies could be consequently seen as complementary to task hierarchies (that include both tasks and actions), each of them holding its role.

Recent approaches put an emphasis on the context of use, sometimes termed as the task world [23]. In this respect, user's characteristics, people and organisations, objects they manipulate and actions they undertake should be considered to develop a usable UI. As a consequence, most task models have a heterogeneous conceptual framework aiming to represent tasks, roles, task decomposition, data flows, goals, actions, objects, and events. The importance given to different models (e.g., environment, domain, and user) varies according to the application type.

Task analysis describes, represents, and analyses computer-supported work, i.e. tasks that are delegated (in part or in full) to the computer. Task analysis for UI should not be considered in isolation but as an activity in a UI development life cycle. When using task models as prerequisites for designing UI, compatibility between the domain-task world and design knowledge is needed. To this respect the following elements are important for a task model:

- Goal hierarchy, because it reflects the goal structure, not necessarily equal or similar to a task structure.
- Task decomposition, because it shows the task structure and the constraints in grouping related tasks in the interface.
- Temporal relationship between tasks, because they show constraints for placing interaction objects.

Other models of the context of use provide critical elements for modelling a task:

- Data flow diagram, as a complementary view that shows the functional requirements of the application in terms of information exchange.
- Command and interaction objects available, because it shows the possible ways to perform an elementary task.

Many of these aspects are considered in an ontology of task models designed by van Welie et al. [23]. Although this ontology was mainly intended to assess concepts in their Groupware Task Analysis (GTA) method [22], this pioneering work has the merit of identifying fundamental concepts of a task model. However, the method used for building this ontology is only partially described in the paper and does not provide any analysis of cross-task models concepts and matching. Dittmar [7] introduced a more general expression of task models where relationships between model elements are expressed as general equations between any level of the model. This generalisation extends the traditional way of characterising a task model as a hierarchy of concepts. But it was neither intended to analyse cross-task models concepts.

3 A Method for Uniforming Task Models

In this section, we present the steps of the method followed to build a uniformed task model from a number of existing individual task models. An individual task model is referred to as any model produced by a specific task analysis method. Uniforming an individual task model is the process of expressing an individual task model into a uniformed model in such a way that its concepts keep always the same form, manner, or degree, and present an unvaried appearance of concepts.

Our method for uniforming task models consists of four major steps: selection of individual task models, identification of the concepts within each model, representation of those concepts into a meta-model, and consolidation of these meta-models into one single meta-model, called *uniformed task meta-model*. Although these steps are presented sequentially, some feedback is still possible at some points.

Considering all existing task models is probably beyond our capabilities. Thus, a selection among task models was operated from the list of task models reported in the CHI'98 workshop on task models, which is supposed to be rather extensive and correct [3]. The following criteria were used:

- The task models should be integrated in a development methodology as a core (e.g., CTTE [17]) or side (e.g., HTA [1]) component and tool supported.
- The task models should be widespread and accepted within the Human-Computer Interaction (HCI) community (e.g., TKS [10]).
- The selected models should be supported by theoretical studies to assess their soundness and experimental studies for effective case studies (e.g., MUSE [15]).
- The selected set should cover the wide scope of disciplines that originated different models (e.g., as CPM-GOMS and NL-GOMS belong to the same family, only one representative member is kept [9]).
- The selected set should also reflect the geographical diversity of origin of different models.

After selecting individual task models, the foundation references of each chosen task model were analysed. Each model was then decomposed into constituting concepts using an entity-relationship method of analysis. The terminology used in original references to refer to concepts was preserved. A definition of each concept is then given. For the sake of concision, only relevant definitions of concepts were retained.

These concepts are then represented into an individual task meta-model, which is made up of entities and relationships expressed according to an entity-relationship-attribute methodology. To support this representation, facilities offered by the DB-MAIN CASE tool [8] have been exploited, as well as their organising rules to ensure a common representation scheme. The ERA formalism was chosen for a main reason: this semi-formal method is recognised for its expressiveness and widely used in both the computer science and the software engineering fields. Furthermore, it is understandable for a non-computer

scientist. However, the ERA formalism does not allow the analyst to represent the concepts semantics, as it is more focusing on the information structure than its underlying semantics. For this purpose, a more formal approach would be helpful, for instance defining each concept with an abstract data type. But the analysis of the cost/benefit ratio of such an approach discouraged us to adopt it. Indeed, such a formal approach would force a formal definition of each concept for each individual task model, thus leading to a huge formalisation effort. In addition, this solving becomes even more compound when the originating documents did not formally define their concepts themselves.

Finally, a uniformed task meta-model is obtained from the individual task meta-models. To build this final meta-model, different intellectual operations were performed. Firstly, a syntactical uniforming was conducted to provide a single way of referring to different concepts where possible. This step implies that concepts having the same definition but different names were uniformed under a same label. For concepts having different definitions, even if they refer to a similar fundamental concept, a semantic uniforming was needed. This step implies the identification of semantic mappings between concepts having different aims and scopes. To maximise the semantic scope of the uniformed task meta-model, the union of the concepts present in each particular task meta-models was preferred rather than the intersection. Indeed, choosing the intersection would produce an "emergent kernel of concepts" common to all methods, but this set may be rather limited. Conversely, the union while keeping commonalities preserves specific contributions of individual models.

To avoid the problem of an all embracing model, some concepts (i.e., entities, relationships, or attributes) were withdrawn from this union for several reasons: the concept is semantically redundant with an already existing concept, the concept is not practically used by the methodology in which the particular task model is defined, the concept does not basically belong to the task model but rather to other models like user, organisation, domain, or presentation model [17]. This reason is motivated by the Separation of concern principle which assumes that only concepts relevant to a similar domain of discourse should be kept in a particular model, thus avoiding mixing different concepts into a single model.

4 Analysing Task Models

From task models reported in the CHI'98 workshop [3], criteria defined in Section 3 resulted in the selection of the following alphabetical list of individual task models: CTT [17], Diane+ [20], ETAG [21], GOMS [2, 4, 9], GTA [22], HTA [1], MAD [19], MUSE [15], TAKD [6], and TKS [11, 12, 10]. Various objectives underlie the elaboration of each individual task model, thus featuring a certain type of representation:

- To inform design about potential problems of usability like in HTA.
- To evaluate human performance like in GOMS.

– To aid design by providing a more detailed task model describing task hierarchy, object used and knowledge structures like in TKS, GTA or CTT.
– To generate a UI prototype like in Adept supporting TKS approach

Due to space constraints, the analysis of only a subset of these selected models is reproduced here. Details can be found in [16].

4.1 The GOMS Model

The Goals, Operators, Methods and Selection rules (GOMS) model was developed as a model for predicting human performance while interacting with a system. The original GOMS model, referred to as CMN-GOMS, is the ancestor [4] of a family of GOMS models that were refinements of the original model [2]. Among them are GOMS Language (GOMSL) [10] and Critical Path Method GOMS (CPM-GOMS) [9]. The first is using a mental programming language and is based on a parallel cognitive architecture, the latter is using a PERT chart to identify the critical path for computing the execution time.

GOMS is, at first, a cognitive model of the user that is used in task design. Therefore, task is not a central concept but rather the method that is describing how tasks are actually carried on. A meta-model describing GOMS concepts is given in Fig. 1.

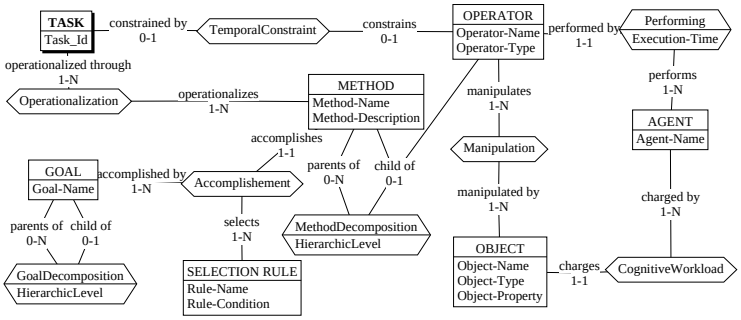

Fig. 1. GOMS task meta-model

A method is a sequence of operators that describes task performance. Tasks are triggered by goals and could be further decomposed into sub-tasks that correspond to intermediary goals. When several methods exist for the same goal, a selection rule, referring to the context of use, is used to choose the appropriate one.

Methods describe how goals are actually accomplished. Higher level methods are describing task performance in terms of lower level methods, operators and selection rules. The lowest level of a GOMS decomposition is the unit task defined as a task the user really wants to perform [9]. Higher level methods are using task flow operators that are acting as constructors that control task execution.

GOMS makes a clear distinction between tasks and actions. Firstly, task decomposition stops at unit tasks. Secondly, actions, termed as operators in GOMS, are specified by the methods associated to unit tasks. Action modelling is varying along different GOMS models and also methods specification.

Operators are cognitive and physical actions the user has to perform in order to accomplish the task goal. Since each operator has an associated execution time (determined on experimental basis), a GOMS model can serve for predicting the time needed to perform a task.

In GOMSL [9], the method is split into steps. Each step is assumed to be a production rule for the cognitive architecture of the user and is assigned 0.05 sec in the execution time estimation. This takes into account the decision time spent for loading a method or returning to the calling one and applies for such control statements as "Method for goal", "Accomplish goal" or "Return with goal accomplished" in the representation above. Therefore, the execution time is calculated as the sum of the statement time and the operator time.

Actions undertaken by the user are specified using external and mental operators. Some special mental operators are flow control operators which are used to constrain the execution flow. Although the granularity is varying according to the purpose of analysis, GOMS is mainly useful when decomposition is done at operational level, i.e. under the unit task level. GOMS cognitive analysis starts after task analysis and it is constrained by the technological options. It is aimed at improving task design with respect to human performance by enabling quantitative predictions like estimation of execution time, learning time, and memory workload.

4.2 The GTA Model

GroupWare Task Analysis (GTA) [22] was developed as a means to model the complexity of tasks in a co-operative environment. GTA takes its roots both from ethnography, as applied for the design of co-operative systems and from activity theory adopting a clear distinction between tasks and actions. An ontology describing the concepts and relations between them was developed in [23]. Fig. 2 graphically depicts five GTA fundamental concepts: task, role, object, agent, and event.

In GTA complex tasks are decomposed into unit tasks [4] and basic tasks [21]. However there is no indication about how this relates to UI design. More recent developments of GTA also include additional techniques like ETAG [21] for decomposition of basic tasks. An attractive feature of GTA relies in its capability of representing co-operative tasks (groupware) by integrating the role concept in the task world, thus enabling representations of task sets for which a role is responsible of and also of organisational aspects like how a role is assigned to different agents.

Although the GTA ontology improves the conceptualisation of the task world the representation it is not appropriately expressed for our purpose. For example, goals and actions are represented as task attributes and not as concepts. This

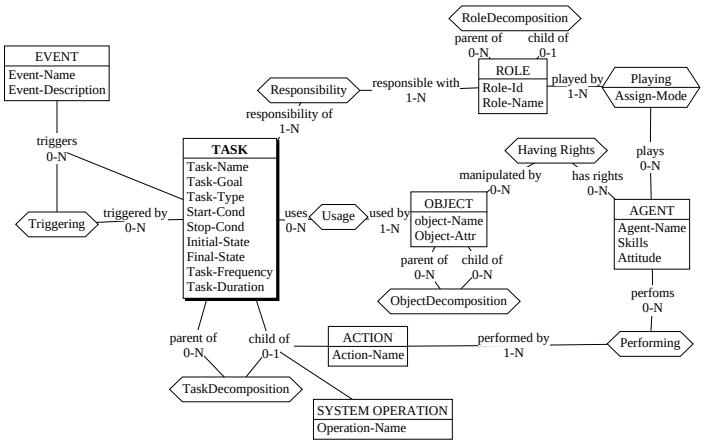

Fig. 2. GTA task meta-model

is somewhat inconsistent with the fact that GTA allows a goal to be reached in many ways.

Also, since the same action could be used in many tasks, actions could be better represented as a concept. This way, object manipulation is represented as a relationship between actions and objects rather than between tasks and objects. Since actions are depending on operational conditions when using different objects different actions might be needed. Another remark concerning the GTA ontology is that constructors are not represented. However, the Euterpe tool used for a GTA task specification allows for specification of constructors in a similar way to MAD (see Section 4.4).

4.3 Concur Task Trees (CTT)

In Concurrent TaskTrees (CTT) [17], CTT constructors, termed operators, are used to link sibling tasks, on the same level of decomposition. This is different from previ-ously described notations where operators describe parent-children relationship thus ranging over all sub-tasks in a task.

CTT uses a tool for editing the task model used to specify tasks, roles, and objects as well as the task hierarchy with temporal operators (Fig. 3). Another feature of CTT is its graphical facility providing means to describe different task types like abstract, co-operative, user, interactive, and application.

CTT provides us with means to describe co-operative tasks: a task model will be composed of different task trees: one for the co-operative part and one for each role that is involved in the task. Tasks are further decomposed up to the level of basic tasks defined as tasks that could not be further decomposed. Actions and objects are specified for each basic task. Application objects are mapped onto perceivable objects in order to be presented to the user.

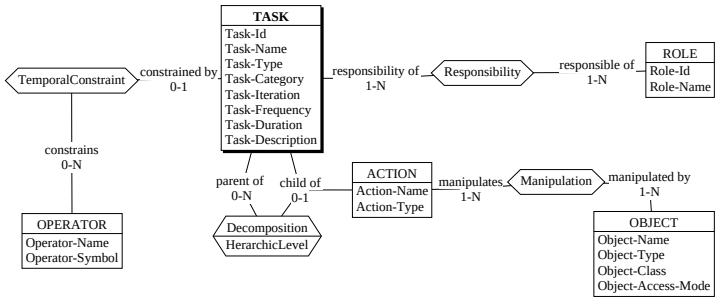

Fig. 3. CTT task meta-model

Another interesting feature of CTT is the specification of both input and output actions that are associated to an object. Object specification is mainly intended for the specification of UI interaction objects (interactors).

CTT modelling tool also provides useful support for task modelling like a model checker and a function to extract enabled task sets. Enabled task sets are consisting of tasks that are enabled in the same time and are produced on the basis of temporal dependencies. This feature is intended as a support for analysis of relationship between the task model and the presentation design.

4.4 The MAD Model

Méthode Analytique de Description de tâches (MAD) [19] aims to provide object-oriented task specifications to support design phases. Main concepts of MAD are: task, action, and structure (Fig. 4).

The structure concept is implicitly represented in the task decomposition relationship and the relationship with the constructor entity. As in GTA, the concept of task is divided into two broad categories: elementary tasks and composite tasks.

Elementary tasks are tasks that cannot be further decomposed. An elementary task contains direct reference to one or several action(s) performed on objects of the domain. The concept of action refers to procedures or methods performed by the system. Composite tasks are tasks that can be further decomposed. A composite task makes direct reference to a single constructor aimed at describing its relationships with its child tasks.

Either composite or elementary, the concept of task, in MAD, is characterised by several attributes. The initial state specifies a state of the world prior to the execution of the task. The final state specifies a state of the world after the task execution. The goal is the objects modifications which the agent want to reach. The goal is explicitly defined here as a subset of the final state.

Though the goal is attached to the task as an attribute, a same goal could be reached by accomplishing different tasks. This would have motivated to make the goal concept into an entity type. The precondition is a set of assertions constraining the initial state. Those assertions must be satisfied prior to the execution of

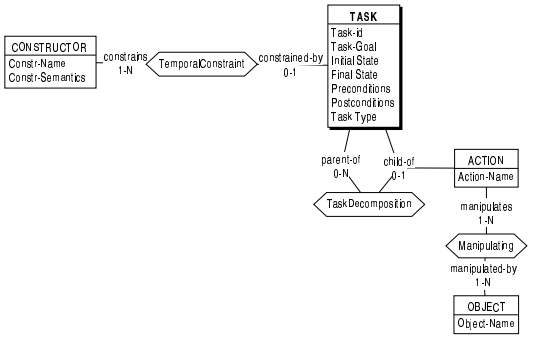

Fig. 4. MAD task meta-model

the task. Preconditions are classified between sufficient conditions and necessary and sufficient conditions. The post-condition is a set of assertion constraining the final state. The post-condition must be satisfied after the execution of the task.

The last central concept of MAD is the structure: it includes the set of decomposition relationships within the task tree and the temporal relationship specification mechanism termed constructor. Constructors operate on all children of a same task. Constructors are classified into two categories: synchronisation operators (SEQ for sequential tasks, PAR for parallel tasks, SIM for simultaneous tasks) and ordering operators (ALT for alternative task, LOOP for iterative tasks, OP for optional task).

4.5 The TKS Model

Task Knowledge Structure (TKS) method [11, 12, 10] manipulates a TKS, which is a conceptual representation of the knowledge a person has stored in her memory about a particular task (Fig. 5).

A TKS is associated with each task that an agent (i.e. a person, a user) performs. Tasks which an agent is in charge of are determined by the role this person is pre-sumed to assume. A role is defined as the particular set of tasks an individual is responsible for performing as part of her duty in a particular social context. An agent can take on several roles. A role can be taken on by several persons.

Even if tasks or TKSs may seem similar across different roles (e.g., typing a letter for a secretary and a manager), they will be considered as different. The "similarity" relationship is aimed to represent this situation.

TKS holds information about the goal. A goal is the state of affairs a particular task can produce. A particular goal is accomplished by a particular task. A goal is decomposed into a goal substructure, which contains all intermediate sub-goals to achieve it.

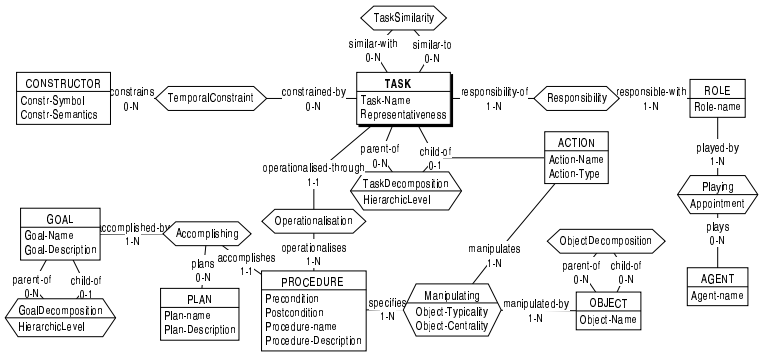

Fig. 5. TKS task meta-model

Goal and task hierarchies have been represented in Fig. 5 as two overlapping substructures, the first decomposing the goal into sub-goals, the latter decomposing tasks into subtasks. Each sub-goal in the goal structure has a corresponding subtask in the task structure and vice versa. The structure is composed either by task decomposition mechanisms and temporal relationship mechanisms termed constructors. Constructors operate on tasks and by association on goals.

A same goal associated can be reached by different sub-goals sequencing. This leads to the concepts of plan. A plan is a particular setting of a set of sub-goals and procedure to achieve a particular goal. As in other models, actions and objects are found at the lowest level of the task analysis. They are the constituents of procedures which operationalise subtasks. One or several procedures can be linked to a same subtask. TKS proposes a production rule system to choose the appropriate procedures depending on the context of use.

Actions are directly related to task tree as they form its leaf level. Actions and objects hold properties of interest. They can be central to the execution of a task, they have typical instances: centrality and typicality of an object or action is always expressed with respect to a task or a procedure that operationalises it. Note that objects are structured into a decomposition hierarchy.

4.6 The DIANE+ Model

DIANE+ [20] formally models a task with three concepts (Fig. 6): operation, sequencing and decomposition.

The operations describe the characteristics specific to a particular task, apart from potential standard actions common to any task such as quit, cancel. This assumes that previously defined standard actions are really common to any task in a particular domain of discourse. Cases where they are not applicable are described.

The described operations, optional or mandatory, are the linked together with operation decomposition and a precedence mechanism. In this decomposition, the triggering condition of an operation is expressed on its sub-operations.

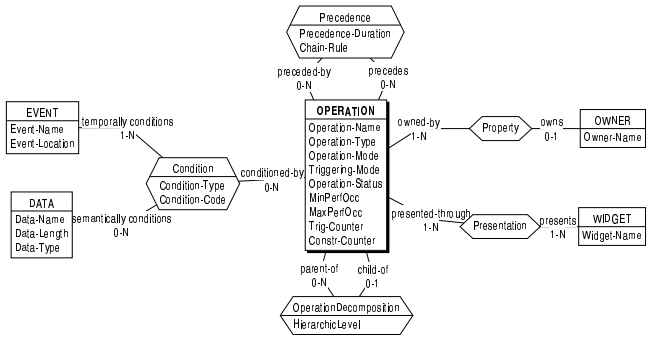

Fig. 6. DIANE+ task meta-model

Pre-condition and post-condition are expressed as either a logical formula on events or on data. Each operation is the property of an owner, which is similar to the agent concept. DIANE+ represents constraints on operation ordering through algorithmic structures resulting from the combination of the basic concepts. For example, an ordered sequence is represented by a simple precedence, an unordered sequence is represented by the required operations and by a lack of precedence. The loop is implicitly represented: an unconstrained user-triggered DIANE+ operation means that the user may execute it as often as she wants. The required choice is represented by an operation with constraint on its sub-operations whereas the free choice is represented by an operation without constraint on its sub-operations. The parallelism is represented through the triggers and a lack of constraint. The loops makes possible to perform the same operation many times in parallel, and several loops can be performed in parallel. The number of constrained sub-operations to perform is specified for each operation.

5 A Uniformed Task Model

Individual task models analysed in the previous section show a variety of concepts and relationships between these concepts. Differences between concepts are both syntactic and semantic. For example, constructors in GTA, MAD or TKS express temporal relationship between a task and its subtasks (although the set of constructors is not identical in all models) while operators in CTT are used between sibling tasks. However, operators used in GOMS have a double semantics. Firstly, they are specifying actions (cognitive and motor) performed by the user. Secondly, in GOMS some operators are also used as syntactic constructions of the language that control the task flow in a way that is similar to a programming language.

In order to access representations that are provided by different task models, a uniforming of concepts is needed. Fig. 7 illustrates common concepts that are used by most models and that are used to build a uniformed task model.

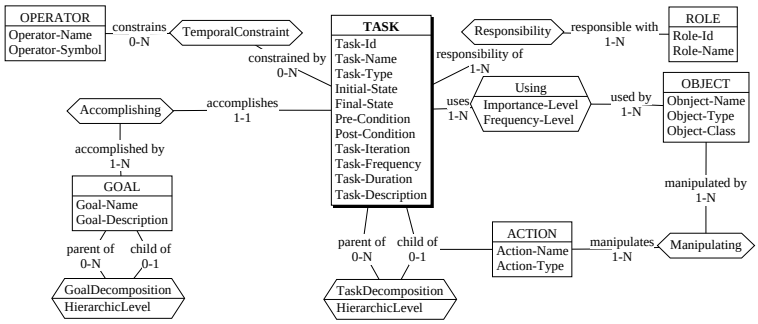

Fig. 7. Uniformed task meta-model

These concepts are believed critical for a task-based design of the UI. In this respect, goal hierarchies enable an identification of presentation units. Operators are expressing temporal constraints between tasks, which in turn impose visibility constraints for grouping the interaction objects. A minimal requirement for dealing with co-operative aspects is role specification, which is done in terms of tasks the user is responsible for. Objects and actions that are performed on them makes possible the detailed modelling of presentation and dialog.

A more detailed rationale for not including concepts of an individual task meta-model in the uniformed task meta-model is given below.

GOMS models assume the preexistence of a given UI. There is no explicit task decomposition but a hierarchic organisation of methods used to operationalise tasks. Cognitive analysis in GOMS is done at unit task level and it requires variable effort of specification which is depending on the level of detail envisioned by the analyst. GOMS is built for the lowest level of task decomposition thus giving no support for the user interface modelling. Rather, the objective is the optimisation of user performance and early evaluation of execution time, memory workload and learning time. Concepts are more related to user modelling than to task modelling.

Event concept is mentioned by few task models. Although GTA and DIANE+ include the event in the task world ontology, they do not model it. Therefore it was not included in the uniformed task meta-model.

GTA defines basic task as being composed of user actions and system operations. However, system operations are relevant for the user only if they are visible in the interface. In this case they are triggering user actions which are related to evaluation of the displayed information. Hence basic tasks are decomposed in actions and this is consistent with task specification from the user point of view. System operations are part of the application model (functional decomposition). Although they could be mentioned for documenting and explaining user actions, they are not part of the task model.

Role concept is used by GTA, CTT and TKS. A role is defined by the task a role is responsible for. Roles are useful to cope with co-operative aspects.

Having a role specification and temporal constraints among tasks it is possible to refine task decomposition in several task trees that show both individual and co-operative aspects. Role decomposition is not relevant for the task model since roles are defined in terms of task sets.

Agent concept is used by GOMS, GTA and TKS although the semantics is differ-ent. As it could be observed from the GTA task meta-model, agent concept implies three relationships: performing actions to manipulate objects, having rights to use certain objects, and playing a role. While the first is drawing on cognitive aspects that could be analysed under the elementary task level, the last two are related to organ-isational aspects. Nor the concept itself neither its relationships to other concepts are useful to this level modelling.

TKS is using both plans and procedures in a similar way with HTA uses plans and GOMS uses selection rules and methods. Although plans are attractive and precise they are informal and could not be checked for proof. Moreover, plans are describing temporal constraints in a procedural way. Adding new tasks lead to rewrite plans that are associated to the next higher level task which may be laborious and error prone. Plans are suitable for early task analysis when information are elicited in an informal but unambiguous way. However they do not provide a representation able to support a (computer-aided) derivation of the UI model.

Table 1. Main features of task models

Features	GOMS	GTA	CTT	MAD	TKS	DIANE+
Task planning	Operator	Constructor (parent)	Operator (sibling)	Constructor (parent)	Plans/ constructor	Precedence/ attributes
Operationalisation	Method		Scenario		Procedure	Operation attributes and constraints
Task tree leaves	Unit task	Basic task	Basic Task	Task	Action	Operations
Operational level	Operators	Actions/ system operations	Actions			
Specification of objects used	Cognitive objects	Object hierarchy	Objects	Objects	Object hierarchy	Data
Cognitive aspects	User performance	User performance/ user knowledge			Knowledge structures	
Co-operative aspects		Roles/agents	Roles co-operative task trees		Roles/ agents	Owners

Like methods, procedures are useful for a detailed specification of the elementary task when there is a need to describe actions performed upon object. However, models should be rather declarative than procedural in order to support successive transforms and to be suitable for using computer tools.

A matching table is presented in Table 1 to evaluate the degree to which task models that were analysed in previous sections fit the uniformed concepts of Fig. 7. The comparison is done on the basis of six features that characterise these task models: task planning, capabilities for operationalisation, lowest level in task decomposition, operational level, cognitive aspects, and co-operative aspects.Regarding the distinction between task and actions an observation should be made. While GOMS, GTA and CTT define an elementary task, other task models do not. For example, the ele-mentary task in MAD is an action. In a similar way, HTA uses the task concept but do not restrict actions to be represented in the task tree.

As it could be observed, none of them satisfies all requirements. On the other hand, optimisation of user performance through a cognitive analysis is not critical for a task-based design of the user interface. From this point of view, task models that are primarily intended as support for evaluation and user training like GOMS are not suitable to support the user interface modelling. Rather, they require an initial design of the interface that is intended to be improved with respect to usability criteria.

This uniformed task meta-model and its underlying semantic mappings will now enable a software tool to accommodate multiple individual task models.

6 A Software Architecture Based on Uniformed Task Models

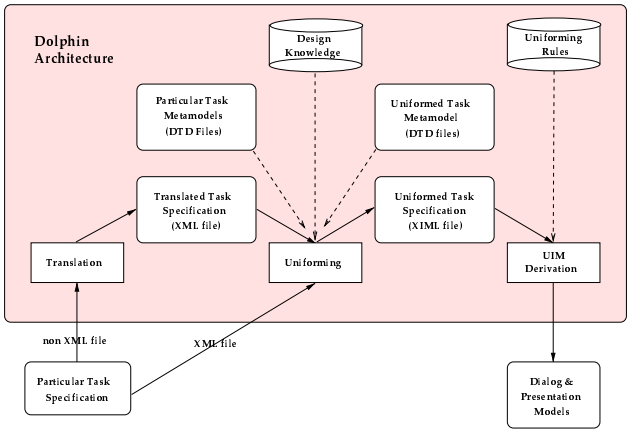

Fig. 8. DOLPHIN software architecture

Domain Ontologies Leading to the Production of Human-computer INterfaces (DOLPHIN) is a UIDA aimed at producing dialog and presentation models from various task models independently of their types. Its architecture is presented in Fig. 8. The input of the system is a task specification file in native format or in a XML-compliant language. If the file is not Extensible Markup Language (XML)-compliant, the file is transformed into an syntactically-equivalent XML-compliant file using the appropriate task meta-model, which has been transformed into a Document Type Definition (DTD) file. At this stage, the file containing the task model is not transformed: only a "XML"ification is generated. XML-compliant files are directly sent to the uniformisation process which maps the concepts of individual task models to the concepts of the uniformed task model. Uniformisation rules, based on the semantic mappings, transform concepts of the individual task model into their uniformed equivalent. For this purpose, uniformisation rules are written as Extensible Stylesheet Language Transformations (XSLT) language [5]. XSLT (XSL Transformations) is a standard way to describe how to transform (change) the structure of an XML docu-ment into an XML document with a different structure. The output of uniformisation process is a uniformed task model expressed in a eXtensible user Interface Markup Language (XIML) file [18]. Design knowledge (e.g., design rules and formalised guidelines) are exploited to produce specification of the dialog and the presentation models. This process and the underlying technology is further detailed in [16].

Main benefits of this software architecture are:

- To allow reusing of existing task tools. Numerous tools (e.g., elicitation tools or graphical editors) supporting particular task models have already proven their qualities. DOLPHIN architecture allows the designer to use the tool that best fits to her needs or her habits while benefiting from a continuous approach.
- To enable incremental design effort. Incremental design is defined as the fact to lead design effort to its end. Some particular task models considered in this paper do not currently support all steps which are involved in a complete development cycle. DOLPHIN facilitates deriving this type of specification until presentation and dialog specification models are obtained.
- To allow reusing existing task models. Task modelling for interactive systems exists and is widespread for a few decades now. The diversity of models has enriched the panel of modelling concepts but it also prevented designers from reusing existing task specifications. Specifications were not accumulated and exploited at a global scale. DOLPHIN permits to consider such practice.
- To enable integration of other task model concepts that could arise in the future. Thanks to DOLPHIN openness, any integration of other task models should be achieved in a cost effective manner.

7 Conclusion

This paper presented a method to transform individual task models into their uniformed counterpart while minimising the loss of information. The stability of the uniformed task meta-model is guaranteed by a correct consolidation of individual task meta-models. If a new task meta-model needs to be consolidated, the uniformed task meta-model should not change, unless more elaborate task characteristics which were not present in the studied task models need to be modelled (Separation of concern).

In this study, only task models have been considered, leaving aside concept relevant to other models (e.g. user, domain). To take this information into account, a similar process may be applied to come up with a more expanded meta-model. In this case, it is no longer a task meta-model as it encompasses aspects of other models.

The main contribution of this paper is a partial solution. Firstly, only a part of the concepts are retained in the uniformed model. The union of the models concepts was chosen rather than the intersection and a selection on the concepts was applied. Secondly, concept semantics is only partially represented. A more formal approach would certainly produce a more reliable representation, but the cost of doing so for each concept for each individual model was judged too high with respect to the expected results. The original DOLPHIN feature described in this paper relates to its capability of accommodating several studied task models as input. For example, any task model stored in CTT's native format or converted into its XML equivalent can be used in DOLPHIN.

References

1. J. Annett and K. Duncan. Task analysis and training design. *Occupational Psychology*, 41:211–227, 1967.
2. L. K. Baumeister, B. E. John, and M. D. Byrne. A comparison of tools for building GOMS models tools for design. In *Proc. Of ACM Conf. On Human Factors in Computing Systems CHI'2000*, pages 502–509, New York, 2000. ACM Press.
3. B. Bomsdorf and G. Swillius. From task to dialogue: Task based user interface design. *SIGCHI Bulletin*, 30(4):40–42, 1998.
4. S.K. Card, T.P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, New York, 1983.
5. J. Clark. XSL: Transformations (XSLT). version 1.0 W3C recommendation. Technical report, W3C, 1999. http://www.w3.org/TR/xslt.
6. D. Diaper. Task analysis for knowledge descriptions (TAKD): The method and examples. In D. Diaper, editor, *Task Analysis for Human-Computer Interaction*, pages 108–159. Ellis-Horwood, 1990.
7. A. Dittmar. More precise descriptions of temporal relations within task models. In P. Pallanque and F. Paternó, editors, *Proc. of the 7th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'00 (Limerick, 5-6 June 2000)*, volume 1946 of *Lecture Notes in Computer Science*, pages 151–158, Berlin, 2000. Springer Verlag.

8. V. Englebert and J-L Hainaut. GRASYLA: Modelling case tools GUIs in metacases. In J. Vanderdonckt, editor, *Computer-Aided Design II. Proc. Of the 3rd Conference on Computer-Aided Design of User Interfaces. CADUI'99 (Louvain-la-Neuve, 21-23 October 1999)*, pages 217–244, Doordrecht, 1999. Kluwer Academics.
9. B. E. John and D. E. Kieras. The GOMS familly of user interfaces analysis techniques: Comparison and contrasts. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351, 1996.
10. P. Johnson. *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering.* McGraw-Hill, Maidenhead, 1992.
11. P. Johnson and H. Johnson. Knowledge analysis of task: Task analysis and specification for human-computer systems. In A. Downton, editor, *Engineering the Human-Computer Interface*, pages 119–144. McGraw-Hill, Maidenhead, 1989.
12. P. Johnson, P. Markopoulos, and H. Johnson. Task knowledge structures: A specification of user task models and interaction dialogues. In *Proc. Of Task Analysis in Human-Computer Interaction, 11th Int. Workshop on Informatics and Psychology (Schraeding, Austria, June 9-11)*, 1992.
13. V. Kaptelinin and B. Nardi. Activity theory: Basic concepts and applications. ACM Press (New York), 2000. CHI'2000 Tutorial Notes vol. 5.
14. B. Kirwan and L. K. Ainsworth. *A Guide to Task Analysis.* Taylor and Francis, London, 2000.
15. K. Y. Lim and J. Long. *The MUSE Method Pfor Usability Engineering.* Cambridge Series on Human-Computer Interaction. Cambridge University Press, Cambridge (UK), 1994.
16. Q. Limbourg, C. Pribeanu, and J. Vanderdonckt. Uniforming of task models in a model-based approach. Technical Report BCHI-2001-4, Université Catholique de Louvain, 2001. Available on request.
17. F. Paternó. *Model-Based-Design and Evaluation of Interactive Applications.* Springer-Velag, 1999.
18. A. Puerta and J. Eisenstein. XIML: Towards a universal user interface markup language. Submitted for publication, 2001.
19. D. Scapin and C. Pierret-Golbreich. Towards a method for task description: MAD. In L. Berlinguet and D. Berthelette, editors, *Proc. Of the Conf. Work with Display Units WWU'89*, pages 27–34, Amsterdam, 1989. Elsevier Science Publishers.
20. J-C Tarby and M-F Barthet. The DIANE+ method. In J. Vanderdonckt, editor, *Computer-Aided Design of User Interfaces, Proc. of the 1st Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996)*, pages 95–119, Namur, 1996. Presses Universitaires de Namur.
21. M. J. Tauber. ETAG: Extended task action grammar. a language for the description of the user's task language. In D. Diaper, D. Gilmore, G. Cockton, and B. Shackel, editors, *Proc. of the 3rd IFIP TC 13 Conf. On Human Computer Interaction Interact '90 (Cambridge, 27-31 August 1990)*, pages 163–168, Amsterdam, 1990. Elsevier.
22. G. C. van der Veer, B. F. Van der Lenting, and B. A. J. Bergevoet. GTA: Groupware task analysis - modelling complexity. *Acta Psychologica*, 91:297–322, 1996.
23. M. van Welie, C.G. van der Veer, and A. Eliens. An ontology for task world models. In *Proc.of the 5th Int. Workshop on Design, Specification and Verification of Interactive Systems DSV-IS'98 (Abingdon, 3-5 June 1998)*, pages 57–70, Vienna, 1998. Springer Verlag.