Revista Română de Interacțiune Om-Calculator 7 (2) 2014, 93-116

Multi-User Interaction Meta-Model

Josefina Guerrero-García[1], Juan González-Calleros[1]

[1]Benemérita Universidad Autónoma de Puebla – Computer Science Faculty
Av. San Claudio y 14 sur s/n, , Puebla, Mexico
E-mail: jguerrero@cs.buap.mx, juan.gonzalez@cs.buap.mx

Abstract. In recent years, there has been a wide interest on how groups of people work together, and on how collaboration might be supported. Many authors, rely on Task Modeling to design collaborative information systems. Task models can be represented at various abstraction levels. When designers want to specify only requirements regarding how activities should be performed, they consider only the main high-level tasks. On the other hand, when designers aim to provide precise design indications then the activities are represented at a small granularity, thus including aspects related to the dialogue model of a user interface (which defines how system and user actions can be sequenced). In this paper a comparative analysis of selected models involving multiple users in an interaction is provided in order to identify concepts which are underexplored in today's multi-user interaction task modeling. This comparative analysis is based on three families of criteria: information criteria, conceptual coverage, and expressiveness. Merging the meta-models of the selected models enables to come up with a broader meta-model that could be instantiated in most situations involving multi-user interaction, like workflow information systems, CSCW.

Keywords: Computer Supported Collaborative Work, Task Modeling, User Modeling, Business Process, Workflow.

1. Introduction

Technology to support groups is rapidly growing in use, some very important trends are: multiple computing platforms, multiple channels, multiple interaction techniques, multiple modalities, multiple environments, and multiple users. In particular, multi-target user interfaces (UIs) (Calvary et al., 1998) explore variations of multiple contexts of use where the context of use is understood as a user interacting with a computing platform in a given environment. Therefore, multiple contexts of use necessarily mean multiple variations of these three dimensions. Among these dimensions, the multiplicity of users has been less researched than the others and has been investigated in different domains ranging from Human-Computer

Interaction (HCI), Computer-Supported Collaborative Work (CSCW) to Collaborative Systems and Workflows (van der Aalst, 2002). Multi-user interaction is hereby referred to as a context of use where multiple users are initiating some interaction and/or receiving the feedback of some previously existing interaction, perhaps in multiple environments.

Multi-user interaction is significant in a certain amount of areas such as: any circumstances where multiple users are involved, whether they are located in the same environment or not (e.g. collaboration, cooperation, competition), where several users are networked in a workflow, where they have individual or shared tasks, where the tasks are multi-user by nature. The problem is that these areas all have their respective understanding and definition of multiple users involved in an interaction. This situation leads to a series of important shortcomings, among them are:

- Lack of understanding: the basic concepts of multi-user interaction modeling are not always well mastered and properly understood, such as the rationale behind their method, their entities, their relationships, their vocabularies, and the intellectual operations involved for modeling these aspects.
- Matching concepts across two different models or more is difficult. It is even likely that sometimes no matching across these concepts could be established.
- Communication among designers is reduced: due to the lack of software interoperability, a designer may experience some trouble in communicating the results of a multi-user interaction model to another stakeholder of the UI development team. In addition, any transition between persons may generate inconsistencies, errors, misunderstandings, or inappropriate modeling.
- Heterogeneousness: these concepts, as they were initiated by various methods issued from various disciplines, are largely heterogeneous.
- Lack of software interoperability: since model-based tools do not necessarily share a common format, they are only restricted to those models which are expressed according to their own, possibly proprietary, format.
- Duplication of research and development efforts: due to the aforementioned differences, different research and development teams may reproduce similar efforts but towards their own format

and terminology, thus reducing significantly the ability to raise incremental research. This shortcoming is particularly important for software development efforts which are resource-consuming.

To address the above shortcomings, we assigned ourselves the next goals:

1. To provide an improved conceptual and methodological understanding of the most significant models involving multiple users and their related concepts.
2. To establish semantic mappings between the different 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, and emergence of transversal concepts.
3. To rely on these semantic mappings to develop a multi-user model editor that accommodates any type of input. This editor should help designers and developers to derive UIs for these multiple users independently of the underlying model. The ultimate goal is to capitalize design knowledge into a single tool and to avoid reproducing identical development effort for each individual model.

The aim of this work is to review the most significant models involving individual or multiple users and their related concepts and to provide a multi-users interaction meta-model that cover the principal characteristics required to work with multiplicity entities playing a role. In the remaining of the paper we present an overview of select models, thus establishing a comparative analysis and the results provided in order to propose a meta-model gathering the concepts identified. Following this, a case study and a tool supporting the meta-model are presented. The paper is wrapped up by summarizing our work, deriving conclusions and addressing future work and challenges.

2. A comparative analysis of Multi-User Modeling Notations

In HCI research, a wide variety of works have been investigated to develop methods for analysing and modeling groupware tasks in multi-user situations. A common definition for a task is “an activity performed to reach a certain goal” (van Welie et al., 1998). A task model is referred to as any

model produced by specific task analysis method. Task models play an important role because they indicate the logical activities that an application should support to reach user' goals.

In this section, we discuss some well-known and widely used notations, examining which characteristics they exhibit and which attributes they cover. It is important to realize that the way we mark a notation is subjective and it is based on our experience. After selecting the individual task models, the foundation references of each chosen task model were analyzed. Each model was then decomposed into constituent concepts using a meta-model. 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 a task meta-model, which is made up of entities and relationships expressed according to an entity-relationship-attribute methodology. Finally, a multi-user task meta-model is obtained from the task meta-models. To build this final meta-model, different intellectual operations have been performed.

Firstly, a *syntactical uniforming* has been 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* is needed. This step implies the identification of semantic mappings between concepts having different aims and scopes. To maximize 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. In order 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 model, organization model, domain model, or presentation model. 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. Task models reviewed are presented in the following sub-sections. Previous work of (Limbourg at al., 2001) was used as a starting point for this research.

2.1. Groupware task analysis

Groupware Task Analysis (GTA) (van der Veer et al., 1996) 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 cooperative systems and from activity theory adopting a clear distinction between tasks and actions.

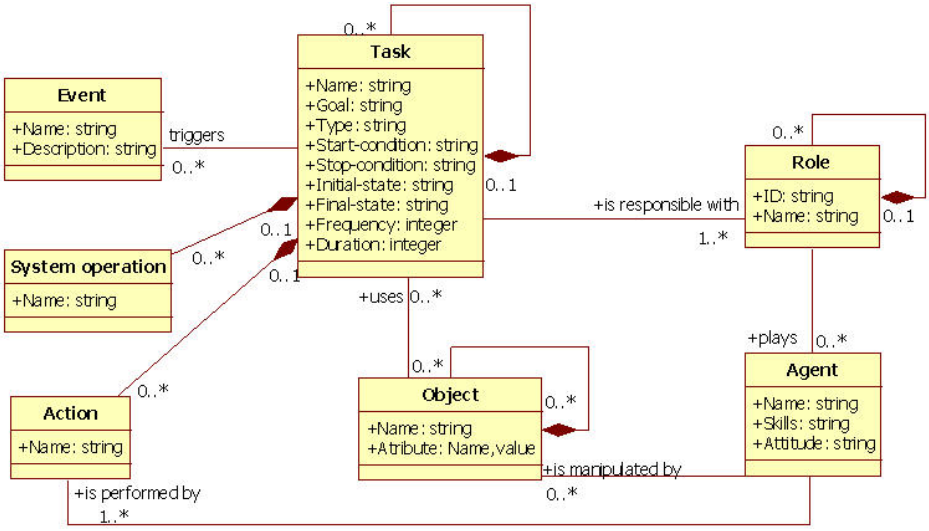

Figure 1. GTA Meta-Model

GTA describes the task world focusing on:

- Agents and roles. Specifying roles and sub-roles that agents play, the relation of responsibility between roles and tasks.
- Work. Involving the decomposition of tasks, the goals and sub-goals, the events that trigger the tasks and the different strategies used to perform them. A task could be performed by an agent or a role.

- Situation. Specifying the objects used in the task world as well as their structure, the history of past relevant events, and the work environment.

Its framework describes a task world ontology that specifies the relationships between the concepts on which the task world is modeled. Based on this ontology a supporting tool to model task knowledge was also developed: EUTERPE [18].

2.2. Task knowledge structure

In Task Knowledge Structure (TKS) method (Johnson et al., 1989; Johnson et al., 2003), the analysts manipulate a TKS, which is a conceptual representation of the knowledge a person has stored in her memory about a particular task. TKS focuses on:

- Roles. A role is assumed to be defined by the particular set of tasks for each an individual is responsible. A person may take on a number of roles and there are tasks associated with each of these roles; or a person could perform similar tasks under different roles.
- Goal structure. It identifies the goal and sub-goals contained within the TKS. The goal structure also includes the enabling and conditional states that must prevail if a goal of sub-goal is to be achieved. In this way the goal structure represents a plan for carrying out the task; the plan is carried out through a procedural structure. A procedure is a particular element of behavior, at the lowest level it can be an action or an object.
- Taxonomic structure. Involves action(s) and object(s) knowledge. This includes the representativeness of the object, the class membership, and other attributes such as the procedures in which it is commonly used; its relation to other objects and actions, and its features (Johnson et al., 2003).

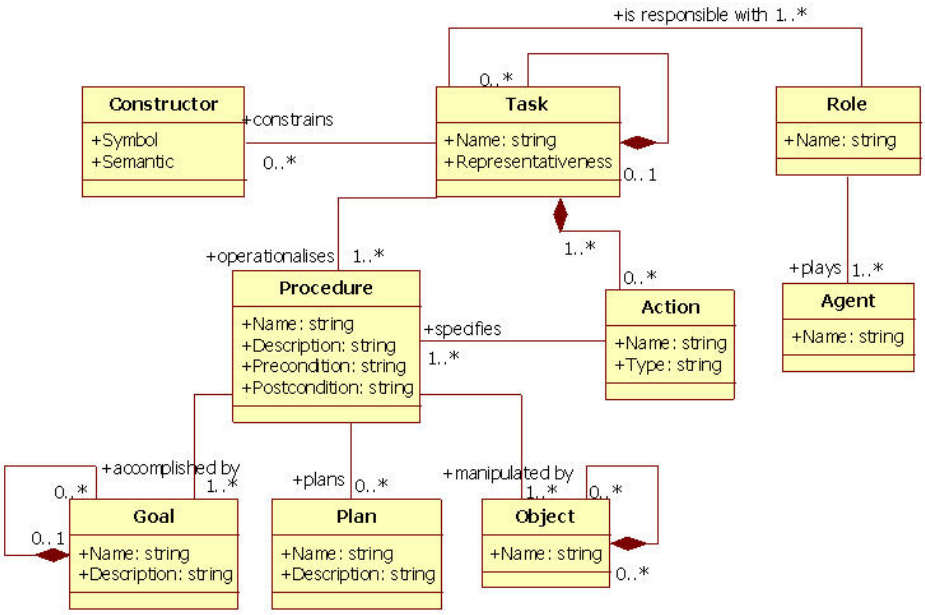

Figure 2. TKS Meta-Model

TKS was not developed on supporting more than one task at a time, but Johnson and Hyde (Johnson et al., 2003) adapted the basic model and extended it to analyze the collaboration work structure. In order to accommodate collaborative tasks, they considered the mechanics proposed by (Pinelle and Gutwin, 2003). Their approach is called Fundamental Knowledge Structures (FKSs). Metaknowledge and mental models constitute the keystone to an FKS for collaboration. It is postulated that there are three different kinds of knowledge that collaborators possess: 1) General knowledge about what makes for an effective collaboration, 2) individual collaborator's specific knowledge of how they will collaborate to complete the task and an understanding of each collaborator's contribution to the task, and 3) collaborator's knowledge of another collaborator's knowledge.

The FKS for collaboration necessarily models high-level knowledge across tasks and consequently is able to generate a set of general requirements for tools to support collaboration across a range of tasks (Johnson et al., 2003).

2.3. ANSI/CEA-2018

ANSI/CEA-2018 (Rich, 2009) is a standard for task model descriptions, which has the potential of significantly improving the usability of computer-controlled electronic products and software interfaces in general. An ANSI/CEA-2018 task model description is an XML document (it is not a graphical formalism) whose syntax and semantics is specified by the standard. The primary use of the XML document is to be interpreted by a device at run-time to guide the user in achieving the described tasks. The key representational features are: Tasks, Input and output parameters, User intent concept, Preconditions and postconditions, Task decomposition, Binding, Grounding, Temporal order, and Applicability conditions.

The concept of task (also called activity, goal, job, action) is at the heart of the standard. Tasks vary widely in their time extent, and some have unbounded time extent. Tasks typically involve both human participants and electronic devices. Some tasks may be performable only by a human being; others may be performed only by an electronic device. Tasks also vary along an abstraction spectrum from what might be called high-level to low-level. A task model defines task classes. A task instance corresponds to an actual or hypothetical occurrence of a task. Input and output parameters represent the data to be communicated with other tasks. The input parameters of a task class should include all data which affects the execution of task instances. The output parameters of a task class should include all data which is modified or created during execution of task instances.

A *user intent concept* is a case frame, consisting of a verb and a set of semantic roles of specified types. The following semantic roles are predefined: agent: Agents are entities that bring about a state of affairs; theme: The theme is whatever is acted upon or most affected or undergoes

motion of some sort, including motion in a metaphorical sense; instrument:

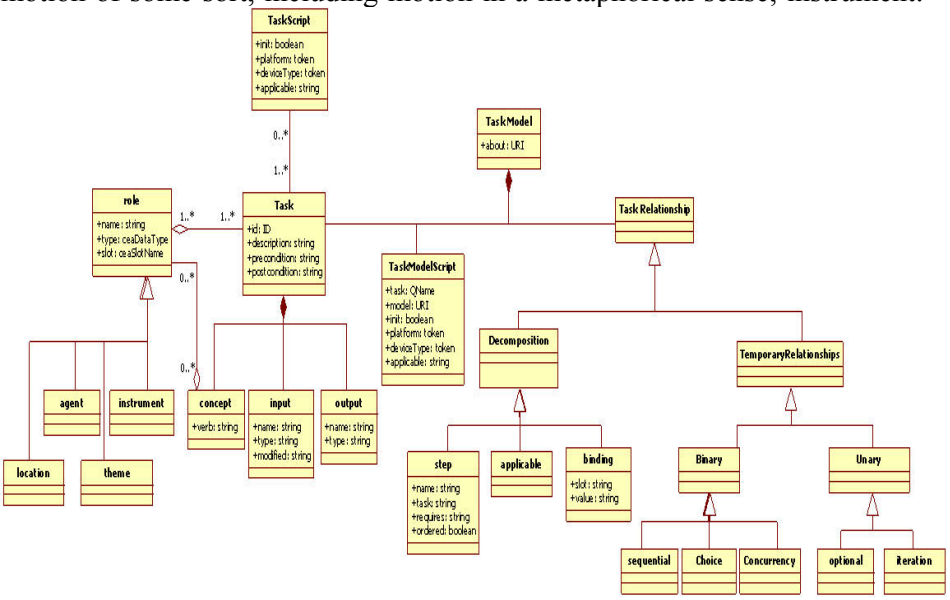

Figure 3. ANSI-CEA Meta-Model

The instrument is whatever is being used to perform the action; and location: Locations are places; they can also serve as the endpoints of paths. The *precondition* of a task is a partial Boolean function which tests whether or not it is appropriate to perform the task at the moment. The *postcondition* of a task is a partial Boolean function which tests whether a just executed task was successful. A task can be *decomposed* into subtasks which are described as steps for executing the task (hierarchy). The *temporal order* between theses subtasks is by default linear (totally ordered), but ANSI/CEA-2018 also supports the specification of partial orders. The data flow between these steps is specified by the *binding* elements in the subtasks definition. In ANSI/CEA-2018, a binding is equality between an input slot of a decomposition step or an output slot of a decomposition goal, and the value of a function with arguments corresponding either to the output slots of steps or the input slot of the goal. *Grounding* is the binding of primitive tasks (those that do not have subtasks) to a script (written in ECMAScript). In ANSI/CEA-2018, a script is an ECMAScript program which may be associated with one or more tasks classes, platforms and device types and whose properties include an applicability condition. For

each decomposition may optionally include an *applicability condition*, which can help the system choose the appropriate decomposition when there is more than one.

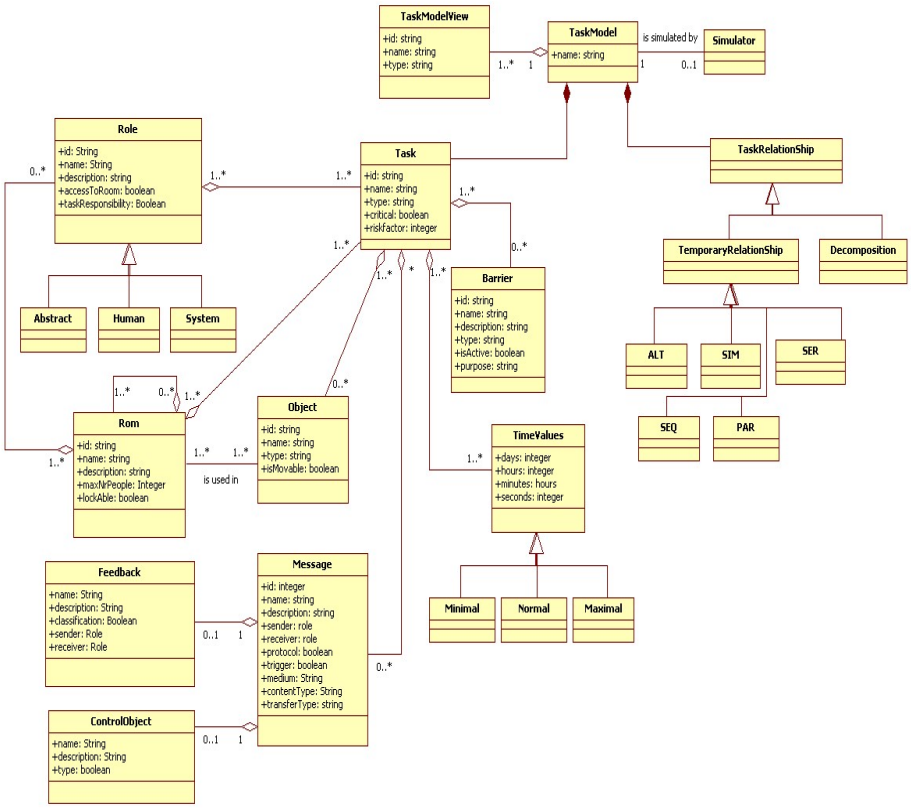

Figure 4. AMBOSS Meta-Model

2.4. AMBOSS

The task models developed with AMBOSS (Giese et al., 2008) describe a hierarchical tree structure of the tasks including the temporal relation between the tasks (formal part of the model) and their description (semi part of the model). On account of this reason that framework shows task models on a semi-formal level. The task model is composed of tasks, rooms, roles and task relationships. Tasks are, notably, described with attributes such as

name and type. The type attribute identifies one of the three basic task types: interactive, system, abstract.

The task also has attributes to determine its duration, the precondition, the severity (indicator for the possible damage that arises from this task), the occurrence, the detection (the likelihood that this failure will be detected), the risk factor (an integer that arises of the multiplication of three values severity, occurrence and detection); and additionally it is possible to make a *risk factor* write protected.

2.5. Diane +

Diane+ (Tarby et al., 1996) is used during the user requirement analysis phase and can guide the design of the user interface. It uses a graphical notation to represent task decomposition as well as temporal and logical relationships among tasks. A Diane+ diagram explicitly indicates whether a task is to be accomplished by the end user (manual), the system (automatic), or a combination of both (interactive).

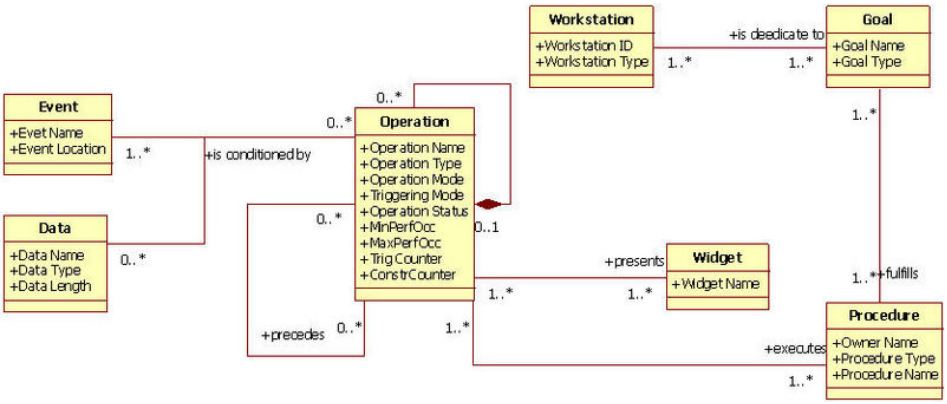

Figure 5. Diane+ Meta-Model

The main characteristics of Diane+ are: the various representations of an interactive application, the abstraction level, the aims and the user´s logics, the dialogue control sharing between man and machine, the adaptation of dialogue to users, and the OPAC data model. There are two important points to be made about the way in which Diane+ models a task. First, the procedures describe only the characteristics specific to an application and do include the standard actions common to most applications, such as quit,

cancel, and so on. This assumes that the supposed standard actions, previously defined, really apply to the application of interest. (If a standard action does not apply, this would be indicated.). Secondly, the described procedures are not mandatory; what is not forbidden is allowed.

2.8. Other Significant Efforts

In ConcurTaskTrees (CTT) (Paternò, 1999) there are five concepts: tasks, objects, actions, operators, and roles. CTT constructors, termed operators, are used to link sibling tasks, on the same level of decomposition. CTT uses a tool (CTTE) for editing the task model used to specify tasks, roles, and objects as well as the task hierarchy with temporal operators. Another feature of CTT is its graphical facility providing means to describe different task types like abstract, cooperative, user, interactive, and application. CTT provides us with means to describe cooperative tasks: a task model will be composed of different task trees: one for the cooperative 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. 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).

UML is a formalism used to design a system with multi-views: use case view, logic view, implementation view, process view, and deployment view. The modeling language UML-G (Rubart et al., 2004) is designed as an extensible UML profile. With this language, groupware developers are supported in modeling their cooperative applications independent from the latter implementation. This supports: explicit modeling of groupware related needs and a shared understanding between developers, which is independent of and thus abstracts from the latter implementation.

UML-G adapts UML in order to describe a groupware application using several stereotypes: <<*shared*>> if used along with an object or a relation, it means that they may be shared during a collaborative session; <<*sharedRole*>> used to identify roles in cooperative sessions; <<*sharedActor*>> used to mark actors in cooperative session that can take

several roles; <<*sharedActivities*>> used to define activities shared in cooperative sessions.

AMENITIES (Garrid et al., 2003) is a methodology based on task models and user behavior for the study and development of cooperative systems. AMENITIES show similarities to other methodologies, they start from a conceptual structure and build behavioral models in the different phases of its life cycle. But it is a less wide methodology in which the most relevant information, related to concepts such as task, role, organization, is included in a unique hierarchical behavior model (UML state chart). It makes easy the information access, and even provides a better understanding of the complete system.

The general schema of the methodology is composed in four modules: 1) Requirements model, where requirements and specifications are collected; 2) Formal model, is an automated analysis, using Color Petri Nets (CPN), which provides the semantics; 3) System design, is the connection point of the methodology with the software development process; 4) Cooperative model is a conceptual model that describes the structure and functionality of a CS.

Goals, Operators, Methods, and Selection rules (GOMS) model, developed by (Card et al., 1983), consists of descriptions of the Methods needed to accomplish specified Goals. Methods are view as a series of steps consisting of Operators that the user performs. A Method may call for subgoals to be accomplished, so the Methods have a hierarchical structure. If there is more than one Method to accomplish a Goal, the Selection rules choose the appropriate Method depending on the context. Today, there are several variants of the GOMS analysis technique, and many applications of the technique in real-world design situations.

GOMS makes a clear distinction between tasks and actions. First, task decomposition stops at task units. Second, actions that in GOMS are termed operators are specified by the methods associated with unit tasks. Action modeling varies depending on the GOMS model and the method 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 experimentally), a GOMS model can help in predicting the time needed to perform a task.

The purpose of Task Object-Oriented Description (TOOD) (Mahfoudhi, 1997), (Ormerod et al., 2004) is to formalize the user task by jointly using

the object-oriented approach and Object Petri Nets. The concepts borrowed from the object approach make it possible to describe the static aspect (Static task model) of tasks and Petri Nets enable the description of dynamics and behavior (Dynamic task model).

The method consists of four steps: hierarchical decomposition of tasks, identification of descriptor objects and world objects, definition of elementary and control tasks, and integration of concurrency. Each task is treated as an instance of a task class identified by a name and an identifier and characterized by a goal, a type (i.e., human, automatic, interactive, and cooperative), the level in the hierarchy, and the total amount of task components. The task body represents a task hierarchy organized using three logical constructors (i.e., AND, OR, and XOR). Each task is then associated with a task control structure (TCS) made up of six classes of descriptor objects that are consumed when the task is carried out and they are aggregated.

2.8. Multi-User Interaction Model Comparison

The task models presented in previous section exhibit a variety of concepts and relationships. The differences between concepts include differences of vocabulary used for the same concept across models. They have, also, different bases of formalization, and scopes. The table below provides the variations between task models; the comparison is based on formalization (this dimension specifies whether a model is based on a formal system or not), role (in order to know if the model uses a role concept.

Task Modeling Operators Comparison Part I

Operators	AMBOSS	ANSI/CEA	CTT	Diane +	GOMS
Decomposition	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy
Sequence	√ SEQ	+- Ordered = true, information passing (Postcondition)	√ Enabling, enabling with information passing	√ Ordered sequence	√ Sequence
Iteration	X	+- MinOccurs+ MaxOccurs	√ Iteration, finite iteration	√ Loop	+- Loop (If, then, else)

Choice	√ ALT	+- Precodition	√ Choice	√ Required choice, free choice	+- Or (If, then, else)
Optionality	+- Barrier	+- MinOccurs+ MaxOccurs	√ Optional	√ Optional	+- Optional (If, then, else)
Interruption	X	X	√ Suspend-resume, disabling	X	+- Interruption (If, then, else)
Concurrency	√ SER	+- Ordered = false	√ Concurrent, concurrent communicating tasks, independence	√ unordered sequence	+- Concurrency (If, then, else)
Cooperation	+- Precondition	X	√ Cooperative	X	X
Parallel	√ PAR, SIM	X	X	√ Parallel	X

Roles are played by agents and are assigned according to organizational rules.), goal (some models make a differentiation between tasks and goals), cooperative aspect (how it supports cooperative work), scope of constructors (expresses the scope of the task elements on which the temporal operators work. The scope can be the parent or the sibling when any temporal operator constraint affects the ordering, respectively, between a father node in the task decomposition and its children or between siblings of the same father), decomposition (show the level of decomposition allowed in the model), operational level (the task decomposition level where actions take place).

Table 2. Task Modeling Operators Comparison Part II

Operators	GTA	HTA	TKS	TOOD	UsiXML
Decomposition	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy

Sequence	√ Seq	√ Fixed sequence	√ Sequence	√ Sequence	√ Enabling, enabling with information passing
Iteration	X	+- Stop rules	X	X	√ Iteration, finite iteration
Choice	√ Or	√ Selective rule	√ Or	√ Choice	√ Deterministic choice, undeterministic choice, inclusive choice
Optionality	+- Start condition	X	X	X	√ Optional
Interruption	+- Stop condition	√ Stop rules	X	√ Interruption	√ Suspend- resume, disabling, disabling with information passing
Concurrency	X	+- Selective rule	X	√ Concurrency	√ Independent concurrency, concurrency with information passing, order independence
Cooperation	√ Cooperation	+- Teamwork	√ Collaboration(FKS extension)	√ Collaboration	√ Cooperation
Parallel	√ And	√ Dual task (time sharing)	√ And	√ Simultaneity	√ parallelSplit (process model)

Similarly, concepts and attributes were treated in order to derive a common ground and propose a multi-user interaction meta-model.

4. A Multi-User Interaction Model

In order to represent group's requirements to coordinate their work among themselves by relying on implicit (e.g., manual, verbal, informal)

communication schemes, it is necessary to addressing (Mandviwalla et al. 1994) criteria for support group interactions, such as the following ones we selected in our work:

- "Support carrying out group tasks" from the individual level continuously throughout the global level: individual, within groups, for the group as a whole, among groups, within organization, and among organizations.
- "Support multiple ways to support a group task": in principle, there should not be unique way to carry out a single group task, but several mechanisms should be offered for this purpose. If a mechanism is no longer available, another one should be selectable.
- "Support the group evolution over time": when the group evolves over time, the workflow definition should be easily maintained and reflected in the system.

Our meta-model (Figure 6) is intended to provide a range of classes, attributes and relationships that cover the majority elements that are encountered when representing multi-user interactions.

The color distinction proved to be useful for the implementation of a workflow editor, to discard classes that were not needed at all for the design of the WfIS (red classes), and to keep an understanding of those concepts that at run-time are to be implemented (yellow and red classes).

Notice that there some classes that are both used at design-time and run-time. An instance of this class can change significantly (the agenda is an example as it changes constantly) thus the use of the red color. An instance of a class that changes moderately (the job definition for the execution of the task is an example as the definition does not normally change on a regular basis) thus the use of the yellow color.

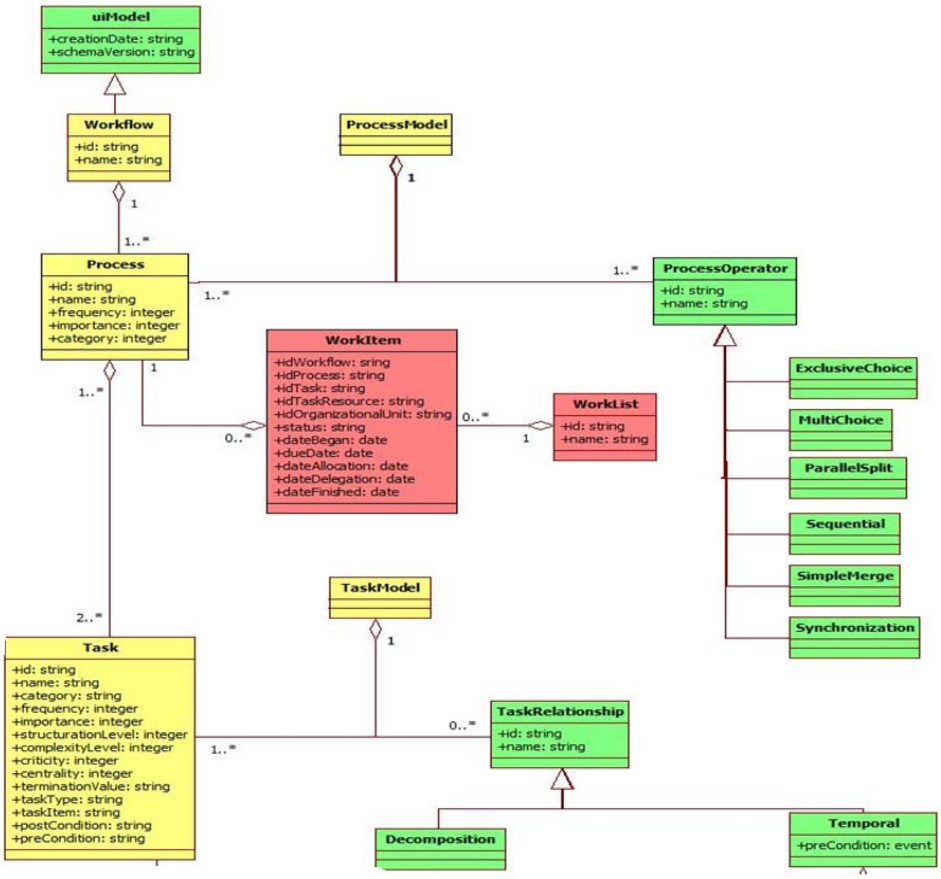

Figure 6 Multi-users interaction meta-model

The task models exhibit a variety of concepts and relationships. The differences between concepts include differences of vocabulary used for the same concept across models. They have, also, different bases of formalization, and scopes. The table 1and 2 briefly illustrates the variations between task models. The comparison is based on *formalization* (this dimension specifies whether a model is based on a formal system or not), *role* (in order to know if the model uses a role concept. Roles are played by agents and are assigned according to organizational rules.), *goal* (some models make a differentiation between tasks and goals), *cooperative aspect* (how it supports cooperative work), *scope of constructors* (expresses the

scope of the task elements on which the temporal operators work. The scope can be the *parent* or the *sibling* when any temporal operator constraint affects the ordering, respectively, between a father node in the task decomposition and its children or between siblings of the same father), *decomposition* (show the level of decomposition allowed in the model), *operational level* (the task decomposition level where actions take place), *tools* (tools that have as basis the task model).

Task operators identified in most task modeling notations are (Guerrero et al., 2008): Decomposition relationships, enabling to represent a hierarchical structure of the task tree. Temporal relationships represent a specification of temporal relationships between tasks. They can be *binary* or *unary*.

Binary relationships are a type of temporal relationships that connects several instances of two different tasks. Enabling relationships specify that a target task cannot begin until source task is finished. Disabling relationships refer to source task that is completely interrupted by a target task. Suspend Resume relationships refer to source task that can be partially interrupted by a target task and after the target task is completed the source task will be concluded. Order Independence relationships are when two tasks are independent of the order of execution. Concurrency with Information Passing relationships are a type of temporal relationships where two tasks are in concurrency execution and passing information between them. Independent Concurrency relationships are a type of temporal relationships where two tasks are executed concurrency but are independent one to each other and there is no information interchange. Enabling with Information Passing relationships specify that a target task cannot be performed until the source task is performed, and that information produced by the source task is used as an input for the target task. Cooperation relationships specify the relationship of cooperation between two or more tasks.

Inclusive Choice relationships specify two tasks that: both could be executed or just one of them or neither of them. Deterministic Choice relationships refer to two source tasks that could be executed but once that one task is initiated the other cannot be accomplished anymore. Undeterministic Choice relationships define the relation between two source tasks in which both task could be started but once one task is finished the other cannot be accomplished anymore. Disabling with Information Passing relationships occur if one task is completely interrupted by another task; and

the information produced in the first task is used as an input for the second task.

Unary Relationships are temporal relationships that connect several instances of the same task. Optional relationships refer to source task that are optional. Iteration relationships indicate source tasks that may be iterated. Finite Iteration tasks indicate tasks that may be iterated n times.

In order to represent group's requirements to coordinate their work among themselves by relying on implicit (e.g., manual, verbal, informal) communication schemes, it is necessary to addressing (Mandviwalla et al., 1994) criteria for support group interactions, such as the following ones we selected in our work: support carrying out group tasks, Support multiple ways to support a group task, support the group evolution over time.

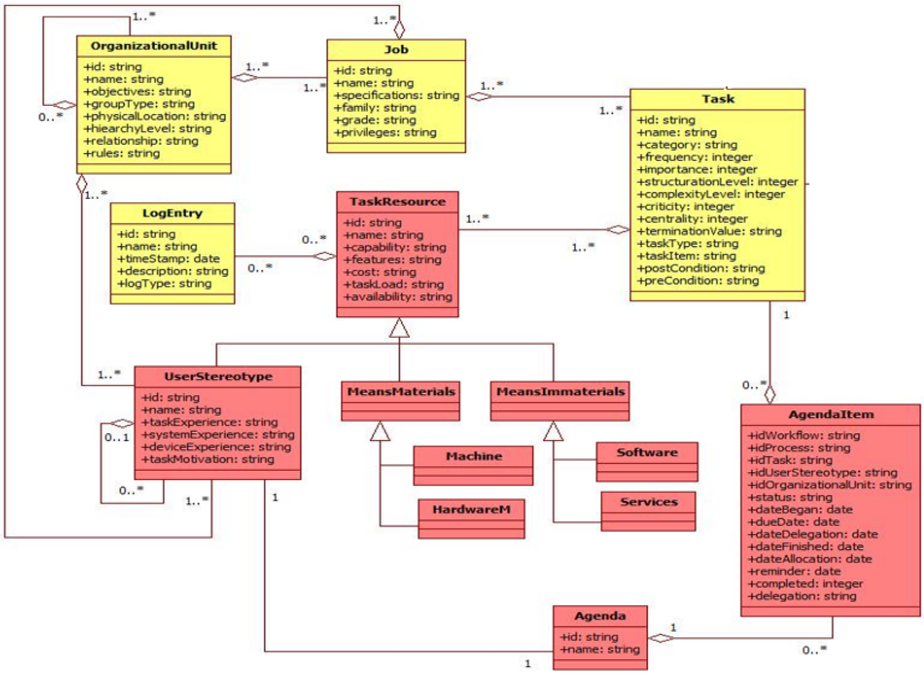

Figure 7 Task interaction meta-model

Figure 7 illustrates the concepts that are used to build a multi-user interaction model. Tasks are organized into processes. A task is decomposed into subtasks and operators are used to link them on the same level of decomposition. A task may manipulate objects through actions. We

introduce the concept of *Job* instead of *role*. Jobs are the total collection of tasks, duties, and responsibilities assigned to one or more positions which require work of the same nature and level. An organizational unit is a formal group of people working together with one or more shared goals or objectives. It could be composed of other organizational units. Resources are characterized thanks to the notion of user stereotype. But a same task could require other types of resources such as material resources (e.g., hardware, network) or immaterial resources (e.g., electricity, power). The agenda is a list of tasks that are assigned to user stereotype. A user stereotype has one and only one agenda and an agenda belongs to one and only one user stereotype. The concept of agenda is useful to cope with the cooperative aspects. We can allocate or offer tasks to user stereotypes through the agenda.

4. Conclusions

In the research literature there is a wide variety of task models with different approaches, it is difficult to consider all in order to elaborate a comparative analysis. To generate our meta-model, we consider those that are supported by theoretical studies, accepted within the Human-Computer Interaction community, and are integrated in a development methodology. Task models analyzed in previous sections show a variety of concepts and relationships. Differences between concepts are both syntactic and semantic.

Syntactic differences cover differences of vocabulary used for a same concept across models. Semantic differences are related to the conceptual variations across models. Semantic differences can be of major or of minor importance. A major difference consists in the variation of entities or relationships definitions and coverage; for instance, a same concept does not preserve a consistent definition across models. A minor difference consists in the variation of expressing an entity or a relationship. For example, constructors in GTA 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.

After the analysis of those task models, a multi-users interaction meta-model was generated in order to cover the principal characteristics required to work with multiplicity entities playing a role. The meta-model applies to identify how tasks are structured, who perform them, what their relative

order is, how they are offered or assigned, and how tasks are being tracked. Moreover, an editor was developed to put in practice the aforementioned model.

Our meta-model tries to cover the principal aspect required to support group work, it include process, tasks, task operators (including collaboration relationship), actions, objects, resources, groups (as an attribute), organizational units, jobs, agendas, goals and rules (both of them as attributes). This work served in different efforts towards the standardization of UsiXML language that is: NexOFRA and W3C MBUI Incubator group. In a future work, we would like to integrate in our comparative analysis other task models that are focused on multi-users interaction. Also, it would be interested to integrate a task analysis part, until now our meta-model is devoted to task model.

References

Bomsdorf, B. and Swillius. From task to dialogue: Task based user interface design. SIGCHI Belletin, 30(4):40-42, (1998).

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (2003) 289–308.

Card, S.K., Moran, T.P. and Newell, A. The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates (1983).

Garrido Bullejos, J.L. AMENITIES: Una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tareas. Ph.D thesis. Granada, España, (2003).

Giese, M., Mistrzyk, T., Pfau, A., Szwillus, G. and von Detten, M. AMBOSS: A Task Modelling Approach for Safety-Critical Systems. Proc. of 7th Int. Workshop on TAsk MOdels and DIAgrams TAMODIA'2008 (Pisa, 25-26 September 2008), Lecture Notes in Computer Science, Springer-Verlag, Berlin.

Guerrero, J., Vanderdonckt, J., Gonzalez Calleros, J.M., FlowiXML: a Step towards Designing Workflow Management Systems, Journal of Web Engineering, Vol. 4, No. 2, 2008, pp. 163-182.

Guerrero, J., Vanderdonckt, J., Gonzalez, J.M., Winckler, M., Modeling User Interfaces to Workflow Information Systems, Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS'2008, IEEE Computer Society Press, Los Alamitos, (2008).

Johnson, P. and Johnson, H., Knowledge analysis of task: Task analysis and

specification for human-computer systems. In A. Downton, (Ed.). Engineering the human-computre interface (pp. 119-144), London: McGraw-Hill (1989).

Johnson, P., Hyde, J., Towards Modeling Individual and Collaborative Construction of Jigsaws Using Task Knowledge Structures (TKS). ACM ToCHI, vol. 10, no. 4, pp. 339-387, ACM Press, (2003).

Limbourg, Q., Pribeanu, C., Vanderdonckt, J., "Towards Uniformised Task Models in a Model Based Approach", in Proc. of 8th International Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'2001, Ch. Johnson (eds.), Lecture Notes in Computer Science, Vol. 2220, Springer-Verlag, Berlin, 2001, pp. 164-182.

Mandviwalla, M., Olfman, L. What do groups need? A proposed set of generic groupware requirements, ACM Transactions on Computer-Human Interaction, Vol. 1, No. 3, pp. 245 – 268, (1994).

Montero, F., López-Jaquero, V., IdealXML: An Interaction Design Tool-A Task-Based Approach to User Interfaces Design, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Chapter 20, Springer-Verlag, Berlin, (2006), pp. 245-252.

Paternò, F., Model Based Design and Evaluation of Interactive Applications. Springer Verlag, Berlin (1999).

Rich, C (2009). Building Task-Based User Interfaces With ANSI/CEA-2018. IEEE Computer, Vol. 42, No. 9, August 2009.

Rubart, J., Dawabi, P., Shared data modeling with UML-G. International Journal of Computer Applications in Technology, vol. 19, nos. 3 / 4, pp. 231-243 (2004).Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M. and Edmond, D. Workflow Resource Patterns. In the 17th Conference on Advanced Information Systems Engineering (CAISE'05). Porto, Portugal. 13-17 June. (2005).

Tarby, J., & Barthet, M. (1996). The DIANE+ Method, International Workshop of Computer-Aided Design of User Interfaces, June 5-7, Facultés Universitaires Notre-Dame de la Paix.

Tauber, M.J., 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 (Eds.), Proceedings of the 3rd IFIP TC 13 Conference On Human-Computer Interaction Interact 90, pp. 163-168, Amsterdam, Elsevier (1990).

van der Aalst, W. and van Hee, K., Workflow Management: Models, Methods, and Systems. THE MIT Press, Cambridge (2002).

van der Veer, G.C., van der Lenting, B.F., Bergevoet, B.A.J., GTA: Groupware Task Analysis - Modeling Complexity. Acta Psychologica 91 (1996) 297–322.

van Welie, M., van der Veer, G.C., Eliens, A., An Ontology for Task World Models. In: Proc. of 5th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'98 (Abingdon, 3-5 June 1998). Springer-Verlag, Vienna (1998) 57–70.

van Welie, M. Task-Based User Interface Design. SIKS Dissertation Series 2001-6. Vrije Universiteit, Amsterdam. Nl.(2000).