Année académique 2008-2009

Working paper 08/28

Towards a Multi-User Interaction Meta-Model

Josefina Guerrero-Garcia, Jean Vanderdonckt,
Juan Manuel Gonzalez-Calleros

Towards a Multi-User Interaction Meta-Model

Josefina Guerrero-García, Jean Vanderdonckt and Juan Manuel Gonzalez-Calleros
Université catholique de Louvain, Louvain School of Management (LSM),
Belgian Laboratory of Computer-Human Interaction (BCHI)
Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
E-mail: {josefina.guerrero, jean.vanderdonckt, juan.m.gonzalez}@uclouvain.be

Abstract

In recent years, there has been a wide interesting in how groups of people work together, and in how collaboration might be supported. Some very important trends are only now being identified, as use of task models for collaborative working. 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. This comparative analysis is based on: 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.

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) [[7]] 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 [[23]]. 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, and coopetition), 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.

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. Analysis on the task models

In HCI research, a wide variety of works have been investigated to develop methods for analyzing and modeling groupware tasks in multi-user situations. A common definition for a task is “an activity performed to reach a certain goal” [[25]]. 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.

2.1. Groupware task analysis

Groupware Task Analysis (GTA) [[24]] 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. 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 [[26]].

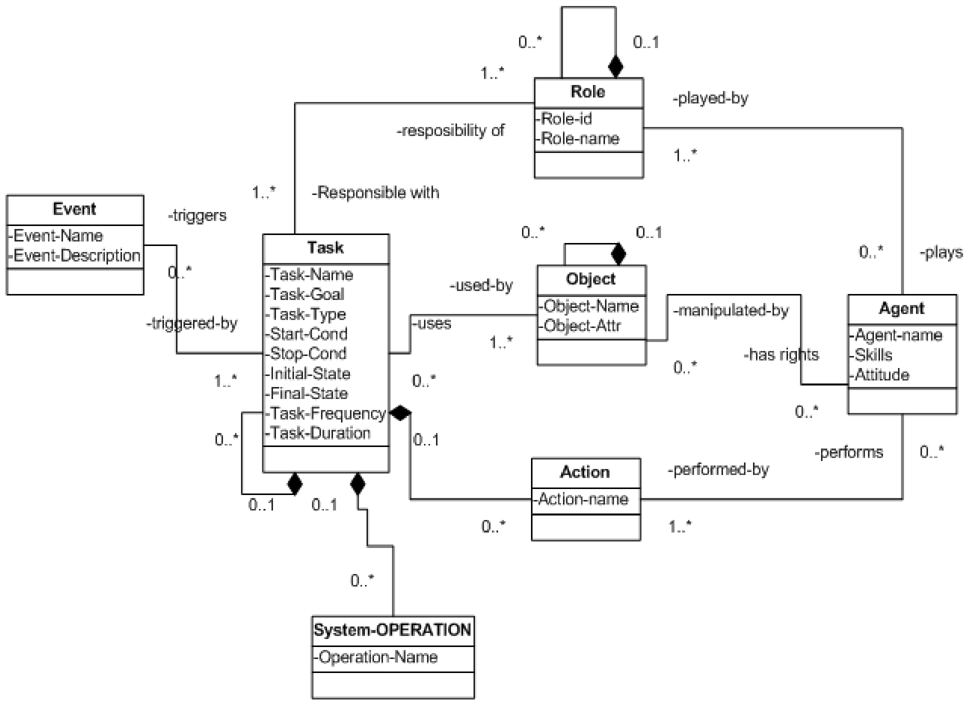


Figure 1: GTA Meta_Model

2.2. Task knowledge structure

In Task Knowledge Structure (TKS) method [[12], [13]], 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 behaviour, 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 [[13]].

TKS was not developed on supporting more than one task at a time, but Johnson and Hyde [[13]] 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 [[19]]. 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 [[13]].

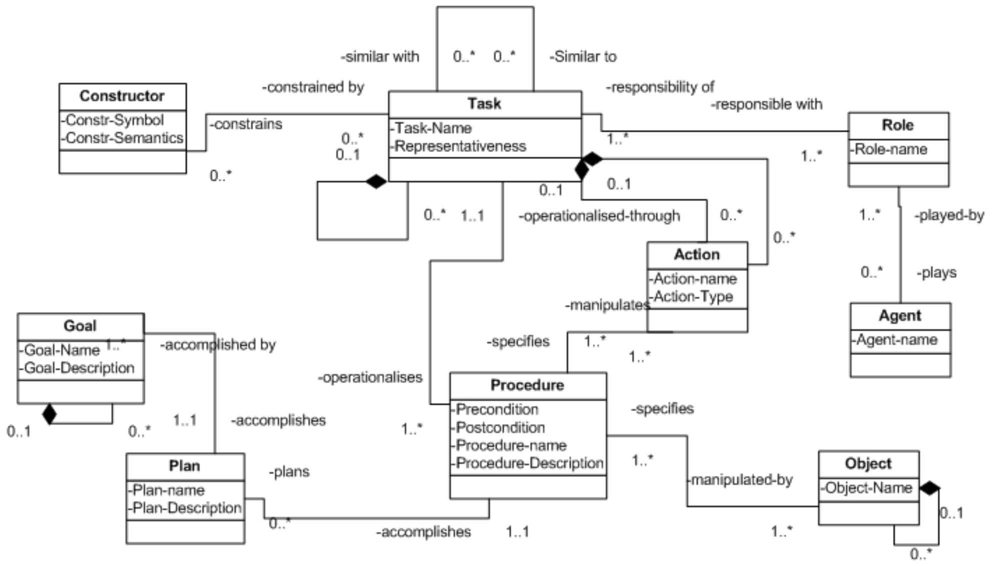


Figure 2: TKS Meta-Model

2.3. ConcurTaskTree

In ConcurTaskTrees (CTT) [[18]] 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).

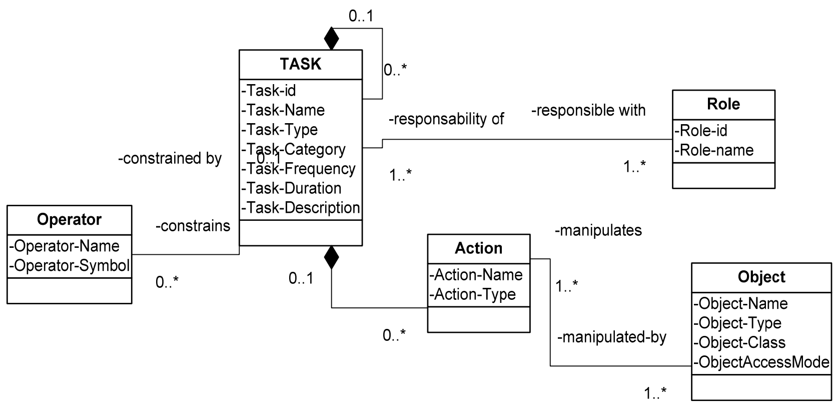


Figure 3: CTTE Meta-Model

2.4. Task Object-Oriented Description

Task Object-Oriented Description (TOOD) consists of an object-oriented method for modelling tasks in the domain of control processes and complex interactive systems, such as those used in air traffic control [15]. The method consists offour steps: hierarchical decomposition of tasks, identification of descriptor objects and world objects, definition of elementary and control tasks, and integration of concurrency (Figure 4). 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:

1. The *triggering* class has four types of events: formal and informal events, events occurring outside and inside the system.
2. The *condition* class contains contextual conditions governing the performance of the task.
3. The *resource* class describes resources (human or system) required for the task to be performed.
4. The *input data* class specifies information items required for performance of the task. To initialize a task, an input transition expresses logical conditions on these data by sending rules and benefits from various checking functions to ensure that all conditions required to perform the task are fulfilled. For instance, the completeness function checks that all input data are available and satisfy related constraints.
5. The *output data* class specifies information items produced by the task performance. To terminate a task, an output transition expresses logical conditions on these data through synchronization rules and benefits from various checking functions.
6. The *reaction* class describes physical and cognitive results resulting from the task performance.

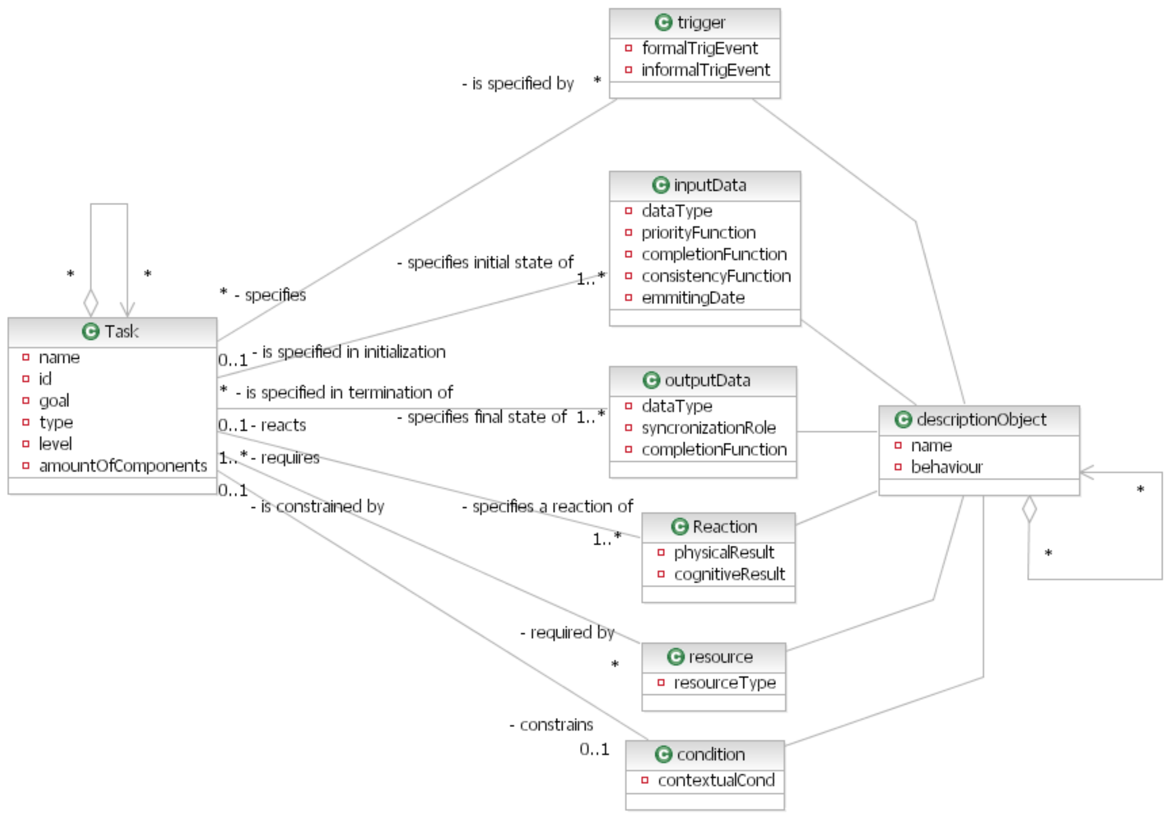


Figure 4: TOOD Meta-Model

The combination of TOOD descriptor objects covers task hierarchy and temporal ordering. TOOD is supported by a graphical editor allowing analysts to specify instances of task classes as well as instances of their related classes.

2.5 Diane

There are two important points to be made about the way in which Diane+ (Figure 5) models a task [4]:

1. 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.)
2. The described procedures are not mandatory; what is not forbidden is allowed.

We note that Diane+ can represent all the constraints of the above specifications. All the algorithmic structures do exist in Diane+, such as ordered sequence, unordered sequence, loop, required choice, free choice, parallelism, default operations, and so on.

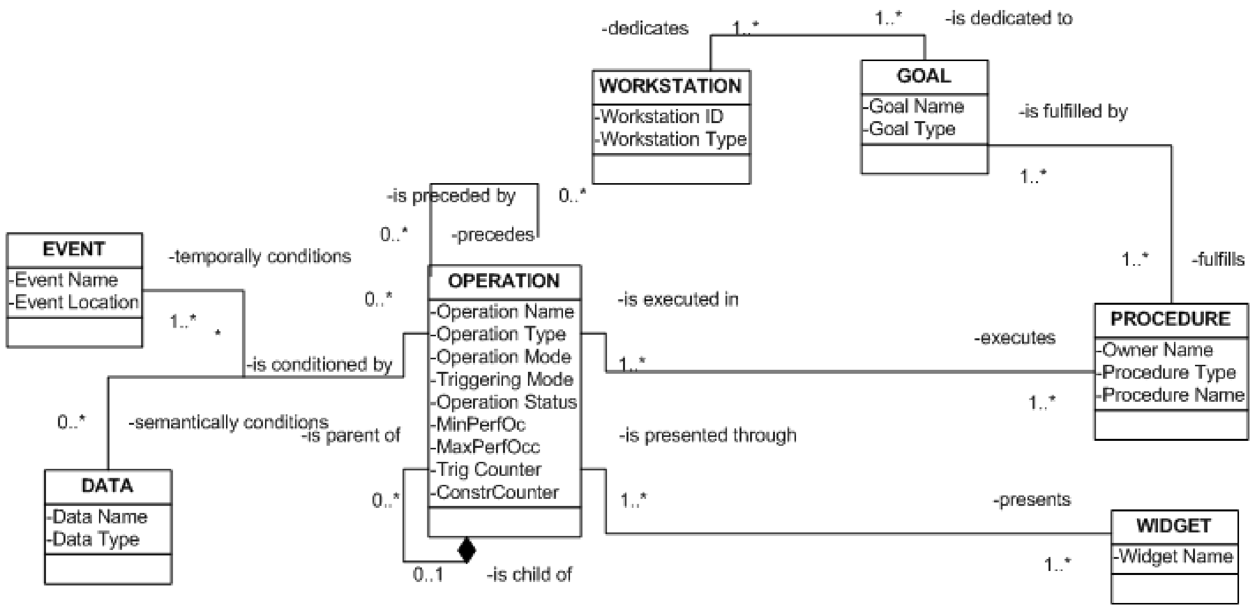


Figure 5: Diane Meta-Model

2.6 HTA

Hierarchical Task Analysis (HTA; [2]) was a pioneering method of task analysis. Itwas primarily aimed at training users to perform particular tasks. On the basis of interviews, user observation, and analysis of existing documents (e.g., manuals, documentation), HTA describes tasks in terms of three main concepts (Figure 6): tasks, task hierarchy, and plans. Tasks are recursively decomposed into subtasks to a point where subtasks are allocated either to the user or the user interface, thus becoming observable. The task hierarchy statically represents this task decomposition. The decomposition stopping criterion is a rule of thumb referred to the $p \times c$ rule. This criterion takes into account the probability of a nonsatisfactory performance and the cost of a nonsatisfactory performance (i.e., the consequences it might produce).

Since the task hierarchy does not contain any task ordering, any task should be accomplished according to a plan describable in terms of rules, skills, and knowledge. A plan specifies an ordering in which subtasks of a given task could be carried on, thus acting as a constraint on task performance.

A plan is provided for each hierarchic level. Although the plan is an informal description of temporal relationships between tasks, it is one of the most attractive features of HTA, as it is both simple and expressive. Plans are very close to textual description or to the activity list of traditional task analysis. One advantage of plans is that they do not create any artificial tasks, as some formal notations force analysts' to do to avoid ambiguous specification.

On the other hand, because plans are informal, it is not possible to apply automatic checking of properties such as consistency and reachability.

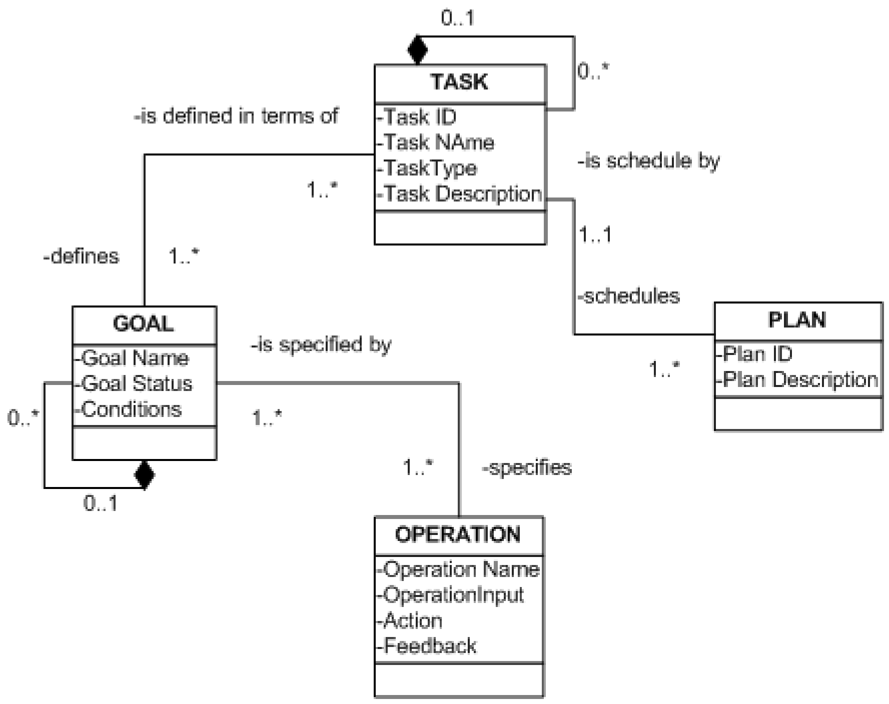


Figure 6: HTA Meta-Model

Any task can be expressed in terms of goals that are reached when the corresponding task is accomplished. Each goal has a status (i.e., latent or active) and conditions to be satisfied. The advantage here in HTA is that goals are independent of the concrete means of reaching them. Therefore, for each goal at any level of decomposition, For each goal, several different operations for reaching the goal can be imagined and specified. Each operation is consequently related to a goal (or goals) and is further specified by the circumstances in which the goal is activated (the input), the activities (action) that contribute to goal attainment, and the conditions indicating the goal has been attained (feedback).

HTA provides a graphical representation of labeled tasks and a plan for each hierarchic level explaining the possible sequences of tasks and the conditions under which each sequence is executed. HTA also supports task analysis for teamwork, as described in [1].

2.7 GOMS

GOMS developed by [8] is an engineering model for human performance to enable quantitative predictions. By incorporating tables of parameter values that rely on a cognitive architecture, GOMS can be used as an engineering approach to task design [3]. The original GOMS model, referred as CMN-GOMS [8], is the root of a family of models that were elaborated later [13], such as GOMSL (GOMS language) and CPM-GOMS (Critical Path Method GOMS).

Although the first uses a “mental programming language” and is based on a parallel cognitive architecture, the second uses a PERT chart to identify the critical path for computing execution time [5].

In GOMS, the concept of a *method* is essential, as methods are used to describe how tasks are actually carried out (Figure 7). A method is a sequence of operators that describes task performance. Tasks are triggered by goals and can be further decomposed into subtasks corresponding to intermediary goals. When several methods compete for the same goal, a selection rule is used to choose the proper one.

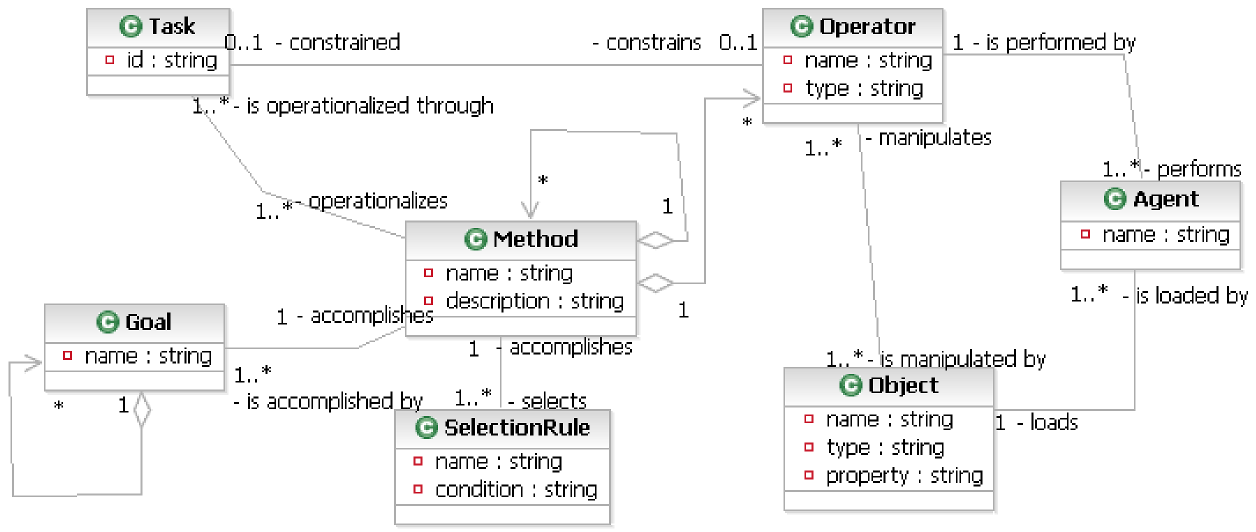


Figure 7: GOMS Meta-Model

Methods describe how goals are actually accomplished. Higher level methods describe task performance in terms of lower level methods, operators, and selection rules. The lowest level of decomposition in GOMS is the unit task, defined by [8] as a task the user really (consciously) wants to perform. Higher level methods use task flow operators that act as constructors controlling task execution.

GOMS makes a clear distinction between tasks and actions. First, task decomposition stops at unit tasks. 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.

Actions undertaken by the user are specified using external and mental operators. Some special mental operators are flow-control operators that are used to constrain the execution flow. Although the granularity varies according to the purpose of the analysis, it is clear that GOMS is mainly useful when decomposition is done at operational level (i.e., under the unit task level).

2.8 AMBOSS

The task models developed with AMBOSS [10] describe the 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 *name* of the task is generally expressed as a combination of a verb and a substantive (e.g., start decent). The *type* attribute identifies one of the three basic task types:

- interactive, involves an active interaction of the user with the system (e.g., selecting a value, browsing a collection of items)
- system, is an action that is performed by the system (e.g., check a credit card number, display a banner).
- abstract, is an intermediary construct allowing a grouping of tasks of different types.

The task also has attributes to determine its *duration* (including its boundaries *minimalDuration* and *maximalDuration* and the time scale used: days, hours, minutes, seconds); the *precondition*, the severity (indicator for the possible damage that arises from this task), the occurrence (the probability that a failure occurs when executing the task), the detection (the likelihood that this failure will be detected), the riskfactor (A riskfactor is an integer that arises of the multiplication of three values severity, occurrence and detection); and additionally it is possible to make a riskfactor write protected (*isWriteProtected)*.

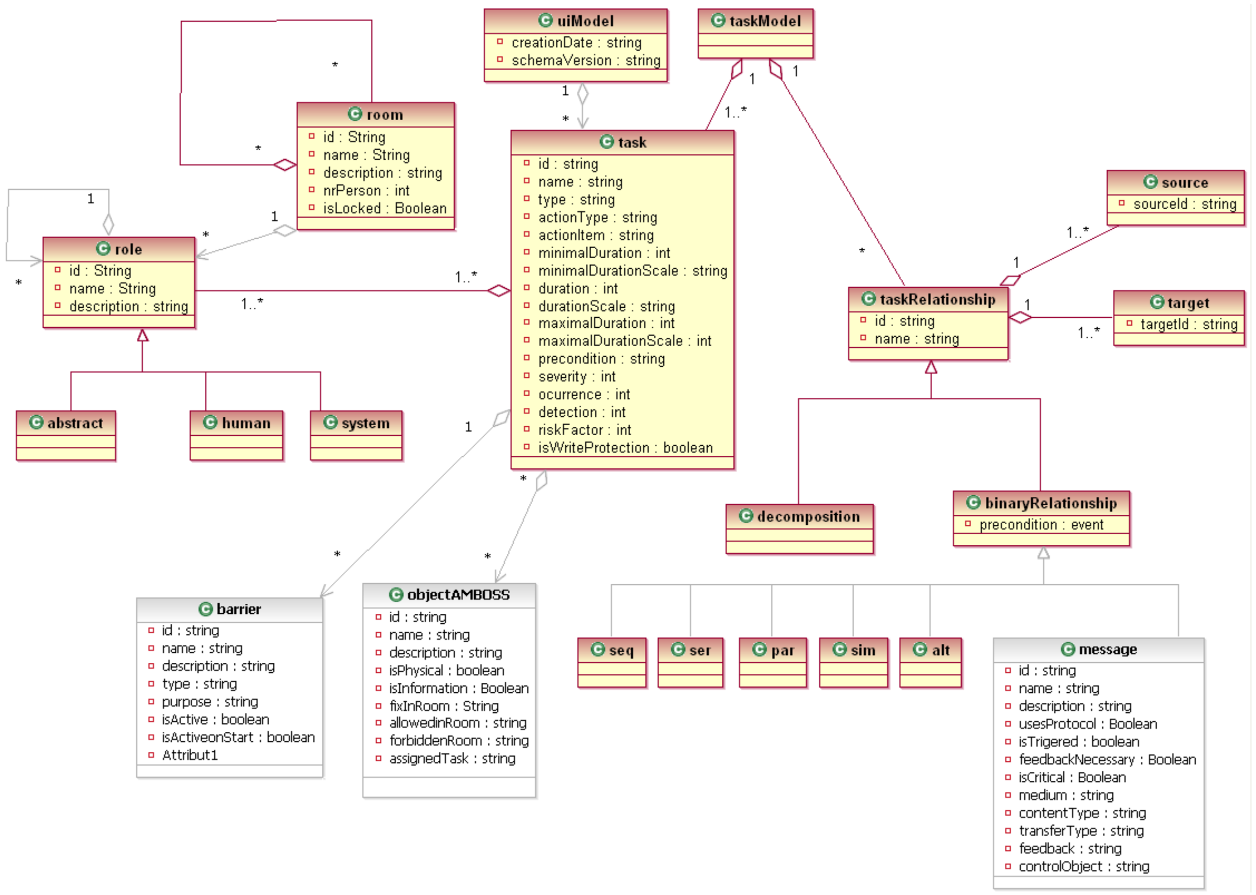


Figure 8:Meta-Model of the Task Model

The following binary *taskRelationship*s are supported in AMBOSS [10]:

- *SEQ*: The subtasks must execute in a fixed sequence from left to right.
- *SER*: The subtasks must execute sequentially but in an arbitrary order.
- *PAR*: The subtasks can start and stop in any order.
- *SIM*: All subtasks have to start in an arbitrary sequence before any task can end. Therefore at least one moment exists where all subtasks are running simultaneously.
- *ALT*: Exactly one randomly selected subtask can execute.
- *ATOM*: The task is the last one in the hierarchy (leaf)

The *decomposition* refers to relationships where a parent task is *decomposed* in subtasks. For each *taskRelationship* the *source* and *target* task must be specified. In addition, *messages* can be transferred from one task to another. A *message* represents the communication between two tasks [10]. It is composed of several attributes like the *name*, the *description*, the *medium* (i.e. electronic, manual, mix), the *contentType* (number, text, graphic, gesture), the *transferType* (synchronous, asynchronous), the *feedback* and the *controlObject* (which ensures the correct delivery of these critical information); also, it contains some flags to determine if the message triggers an action (*isTriggered*), uses a specific protocol *(usesProtocol*), expects feedback (*feedbackNecessary*), is critical (*isCritical*).

A *barrier* determines the protective mechanism of a certain task. It provides the correct execution of the task, which it is assigned to. The assignment is being classified either in safe or unsafe assignment. A barrier consists of an *id*, a *name*, a *description*, a *type* (physical, check, diagnosis, supervision, warning, equipment, procedure, knowledge) and a *purpose* (prevention, control, reduction). Barriers can be active or not, *isActive*, can be activated when the task is started, *isActiveonstart*.

An *objectAMBOSS* is a unity, which is physically available and operating in a running system. An *objectAMBOSS* contains information, which is represented by attributes. It can be either a physical (*isPhysical)* or an informational one (*isInformation)*. Tasks access such objects directly. Also, the objects could be *fixInRoom*, being *allowedinRoom*, *being forbiddeninRoom*, or has a series of *assignedTask.* A *room* denotes the spatial position of the role involved in the execution of the task. A *room* has different static properties, a unique *id* and *name*, a *description*, a maximum number of persons (*nrPerson)* and a flag that indicates, whether this room isLocked or not, i.e. this area is not available.

Finally, a *role* describes the different actors within the task model. This role is an abstract entity. The expert who starts modelling and does not know at that time who is going to execute the particular task uses this kind of roles. They are responsible for the correct handling of the tasks they are assigned to. There are three predefined roles, the *abstract role* (this means we do not or can not assign to just the task to one role) the *human* and the *system* role. Roles execute tasks and they perform their task in a *room*.

4. A multi-users 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 & Olfman [[14]] 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 9) is intended to provide a range of classes, attributes and relationships that cover the majority elements that are encountered when representing multi-user interactions.

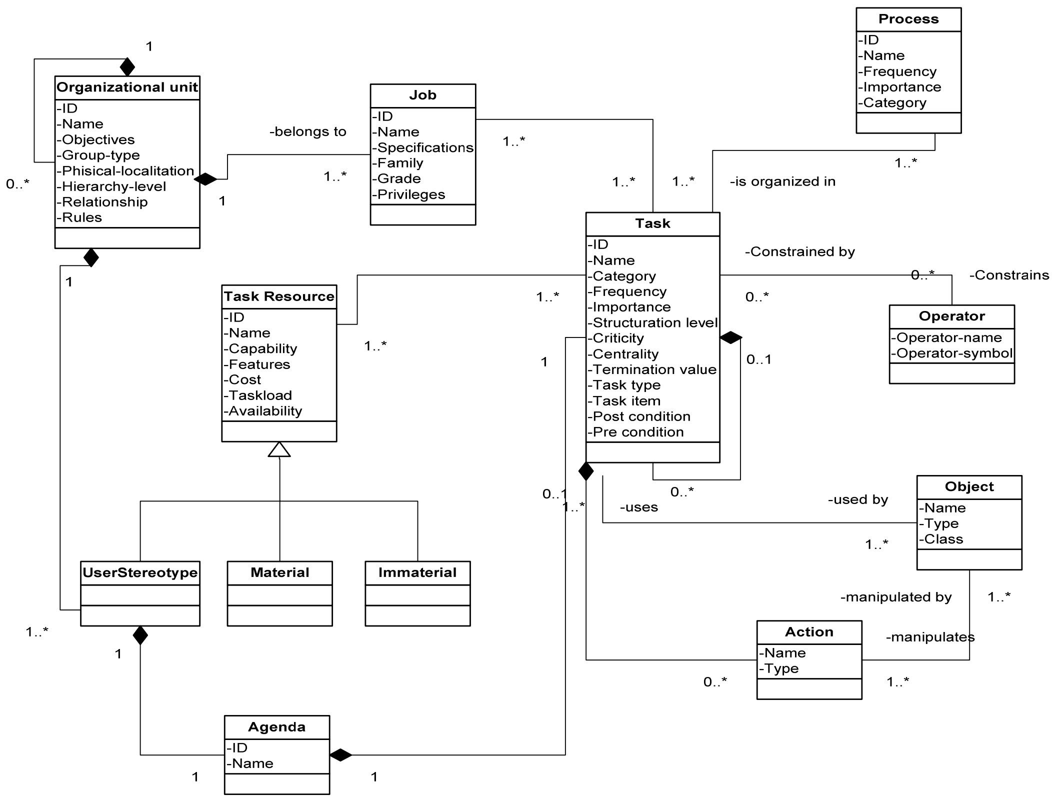


Figure 9: Multi-users interaction meta-model

In this meta-model, tasks are organized in a high-level of abstraction called *processes*. A process consists of a number of tasks and a set of relationships among them. The definition of a process indicate which tasks must be performed and in what order. A *task* can be: user, abstract, interaction or application task. It is decomposed into subtasks to consider hierarchical structure of a task tree; *operators* are used to link them on the same level of decomposition. LOTOS operators are used here for enabling, disabling, concurrency, and synchronization between tasks. 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.

The job concept allows assembling tasks under a same umbrella in a way that is independent of individual resources in the organization unit. In this way, several individuals could play a particular job, and jobs could be interchanged dynamically. Typically, only resources and their roles within organizations are modeled in most task models, we consider that the place where the tasks are executed is an important aspect in the environment where the collaboration is developed. Thus, we introduce the concept of organizational *unit*, it is a formal group of people working together with one or more shared *goals* or objectives. It could be composed of other organizational units. We introduce the concept of organizational unit because across organization, asynchronous and synchronous sharing work objects and coordination are key issues. *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.

5. Tool support and case study

As any meta-model needs a concretization to be manipulated, an editor was developed to represent the above concepts. With this editor, we try to answer the questions: what to do, how to do that, who will do it, where it will be doing, and how tasks will be allocated.

The case study analyzes how people organize the program of small conferences by using a review tool. To organize a conference it is necessary identify the tasks, the jobs and resources which are involved. These tasks were assigned to each job taking into account the role that each of them plays in the workflow. The principal tasks and the job in charge of develop them are:

- Organizer: find the program committee, prepare the call for paper, distribute the call for paper, install conference tool, configure conference tool, find keynotes, organize submission, organize reviewing process, assign papers to reviewers, ask reviewers for preferences, organize discussion, take final decision, announce results, edit proceedings, compose program, edit final proceeding.
- Reviewer: download paper assigned, review paper, give feedback (accept or reject papers).
- Author: submit paper, prepare final submission, attend the conference and present the paper.

Also, it is necessary specify the resources that are available for doing the work and where they are working, i.e. the organizational unit. Author resources are named as A-1, A-2, and A-3 because we do not know who will be interested in submit a paper.

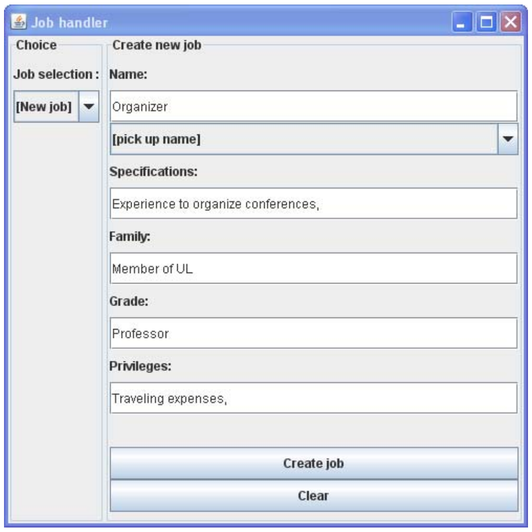
Table 1: Resource and organizational unit identification

Resource	Job	Organizational unit
Chloé Lambin	Organizer	UL
Jacques Khelil	Organizer	UL
Ellen Martin	Organizer/Reviewer	UL
Angela Buker	Organizer	UL
Steve Geller	Reviewer	Reviewer's university
Rachel Walsh	Reviewer	Reviewer's university
Dylan Leitz	Reviewer	Reviewer's university
A-1	Author	Author's university
A-2	Author/Reviewer	Author's university
A-3	Author	Author's university
Review system	-	UL

As we established before, it is possible that a resource have different jobs, in this case a reviewer can be also an organizer or an author.

Within the editor, we can add the different jobs and resources (workers) involve in the organizational conference. Figure 10 shows other attributes of the job, as the specification, family, grade, and privileges. Also, we can have other attributes for the resource (worker) as the level of experience, hierarchy level (Figure 11).

After, we can represent with rectangles each organizational unit, i.e. UL, Reviewer's university, and Author's university. Also we have a representation (small rectangles) of each job assigned to each organizational unit, at the same time, is possible to have the number of resources (small dots with a number) available for develop tasks.



Job selection : [New job]

Name: Organizer

[pick up name]

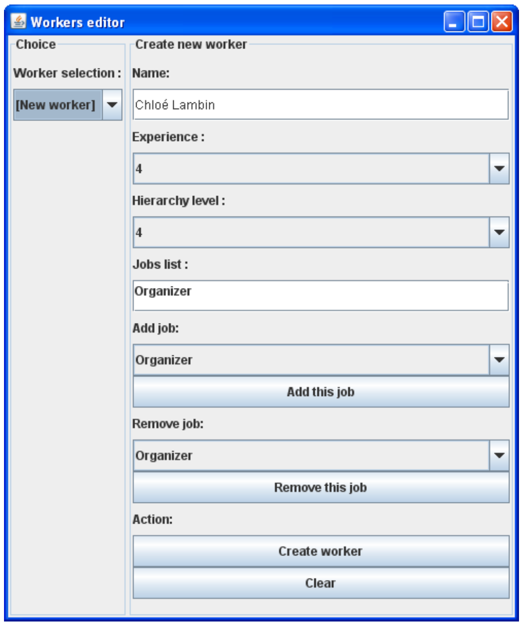
Specifications: Experience to organize conferences,

Family: Member of UL

Grade: Professor

Privileges: Traveling expenses,

Figure 10: Specification of jobs



Worker selection : [New worker]

Name: Chloé Lambin

Experience : 4

Hierarchy level : 4

Jobs list : Organizer

Add job: Organizer

Remove job: Organizer

Action:

Figure 11: Specification of resources

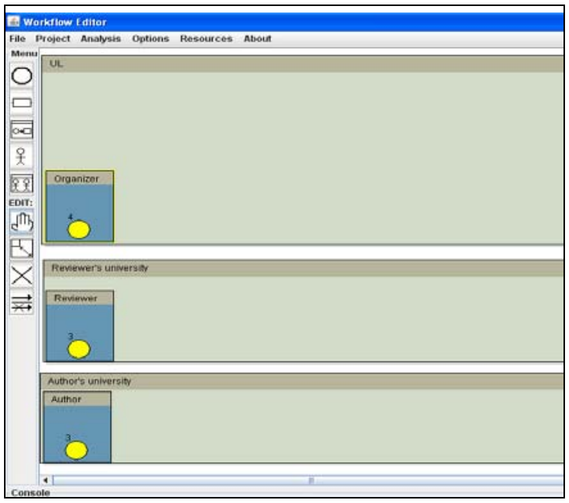


Figure 12: Representation of jobs and resources

After the identification of tasks, jobs, and resources, it is possible to assign tasks to resources applying the workflow resources patterns. This assignation was elaborated after the analysis of the characteristics (qualifications, skills, abilities, experience, and hierarchy level) of each resource and considering the requirements of each task. For instance, *Install conference tool* task was assigned to *Ellen Martin* because she is computer science engineer.

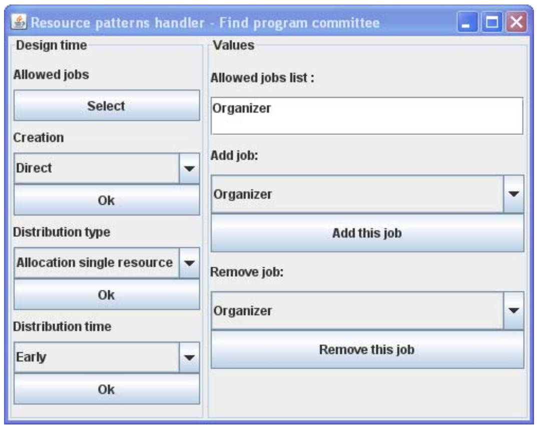
Table 2: Assigning tasks to resources.

Task	Job	Resource	Pattern
Find the program committee	Organizer	Chloé Lambin	Direct allocation
Prepare the call for paper	Organizer	Jacques Khelil	Capability based
Distribute the call for paper	Organizer	Jacques Khelil	Retain familiar
Install conference tool	Organizer	Ellen Martin	Capability based
Configure conference tool	Organizer	Ellen Martin	Retain familiar
Find keynotes	Organizer	Chloé Lambin	Capability based
Submit paper	Author	A-1, A-2, A-3	Deferred
Organize submission	Organizer	Jacques Khelil	History based
Organize reviewing process	Organizer	Chloé Lambin	Capability based
Assign papers to reviewers	Organizer	Angela Buker	Hierarchy level based
Download paper assigned	Reviewer	Steve Geller, Rachel Walsh, Dylan Leitz, Ellen Martin, A-2	Direct allocation
Review paper	Reviewer	Steve Geller, Rachel Walsh, Dylan Leitz, Ellen Martin, A-2	Direct allocation
Ask reviewers for preferences	Organizer	Chloé Lambin	Retain familiar
Give feedback (accept or reject papers)	Reviewer	Steve Geller, Rachel Walsh, Dylan Leitz, Ellen Martin, A-2	Capability based
Organize discussion	Organizer	Chloé Lambin	Direct allocation
Take final decision	Organizer	Angela Buker	Capability based
Announce results	Organizer	Jacques Khelil	Distribution by allocation single resource
Prepare final submission	Author	A-1, A-2, A-3	Deferred
Edit proceedings	Organizer	Angela Buker	Capability based
Compose program	Organizer	Chloé Lambin	History based

Attend the conference and present the paper	Author	A-1, A-2, A-3	Deferred
Edit final proceeding	Organizer	Angela Buker	Capability based

This representation is also possible in the workflow editor. First, we select the job (Figure 13) and the type of workflow resource pattern, after we select the resource (worker) that will develop the task. In this case study, we show the workflow resource patterns that can be used during the design phase of the workflow.

Once the identification of resources, organizational units, jobs, and the assignation of tasks to resources were specified, we can started to model the process and task model.



Allowed jobs
Creation: Direct
Distribution type: Allocation single resource
Distribution time: Early
Allowed jobs list : Organizer
Add job: Organizer
Remove job: Organizer

Figure 13: Select of job and workflow resource pattern

The processes (i.e. the way that tasks are grouped) are modeled using Petri net notations [[23]], modelling a process definition in terms of a Petri net is rather straightforward: *tasks* are modelled by *transitions*, *conditions* are modelled by *places*, and *cases* are modelled by *tokens*. In the process dimension, building blocks such as the AND-split, AND-join, OR-split, OR-join are used to model sequential, conditional, parallel and iterative routing. Figure 14 represent a partial view of the tasks involved in the case study; each task is placed according to the organizational unit where it is developed. Petri nets have been used to model and analyze all kinds of processes with applications ranging from protocols, hardware, and embedded systems to flexible factoring systems and user interaction. There are several reasons for using Petri nets: because the semantics of the classical Petri net and several enhancements (color, time, hierarchy) have been defined formally, they are intuitive and easy to learn due to their graphical nature, they accommodate multiple tokens at the same time, and this means that, for instance, several documents could be processed simultaneously in different states.

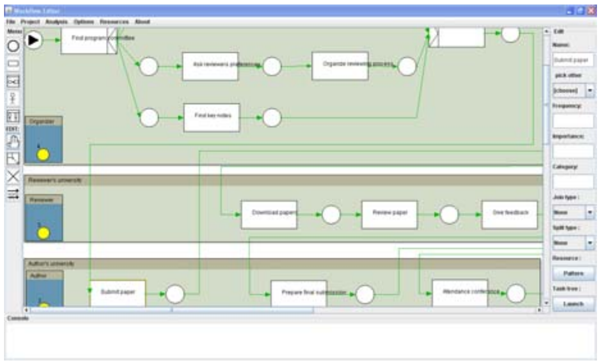


Figure 14: Process representation

For each task, a task model can be designing to specify the decomposition in subtasks (i.e. hierarchical structure). This part of the editor is based on CTT [18] and IdealXML [[17]]. The hierarchical decomposition allows the user to know which tasks need to be carried out in order to realize its parent task. Temporal operators specify a temporal ordering of tasks. Those tasks being exclusively those defined as children of the main root task, no other task is involved here.

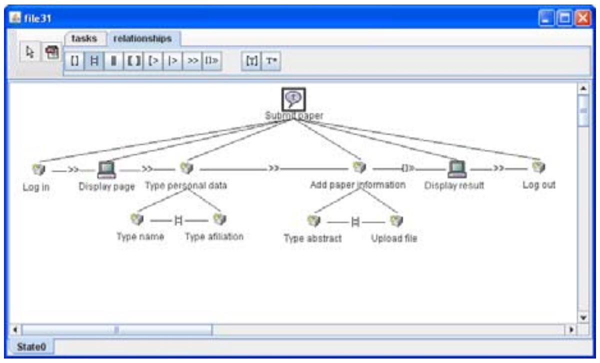
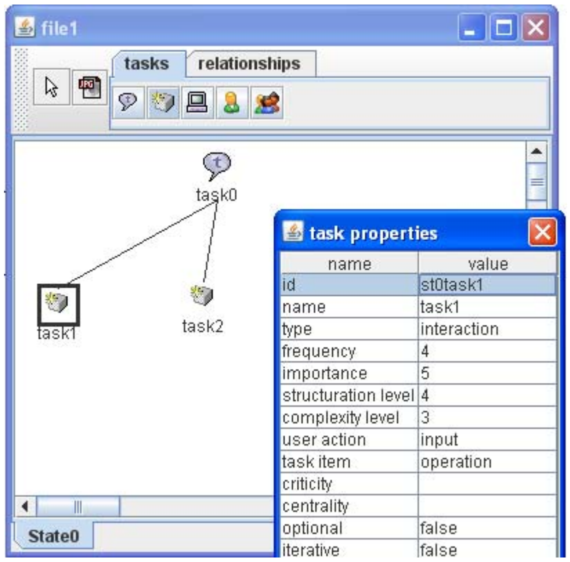


Figure 15: Task model

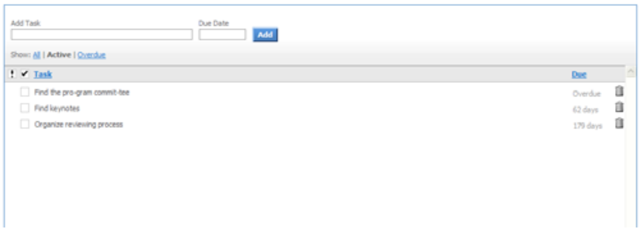
In task model part, the properties of the task can be added.



name	value
id	st0task1
name	task1
type	interaction
frequency	4
importance	5
structuration level	4
complexity level	3
user action	input
task item	operation
criticity	
centrality	
optional	false
iterative	false

Figure 16: Task properties

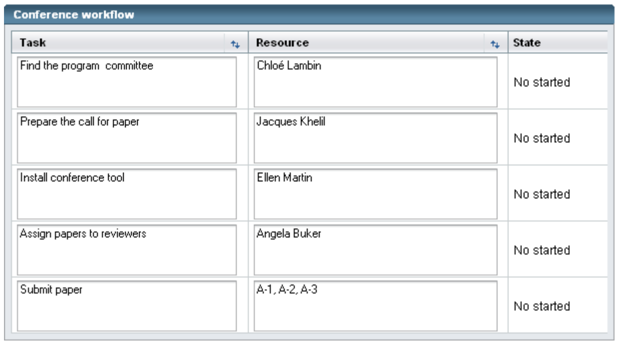
Following with the collaboration among resources, by the agenda (Figure 17) a resource is notified about the possible tasks that are allocated or offered to him. S/he can decide to develop the task or delegate it to another resource that fulfills the requirements.



!	✔	Task	Due
	☐	Find the pro-gram commit-tee	Overdue
	☐	Find keynotes	62 days
	☐	Organize reviewing process	179 days

Figure 17: Agenda

In order to have control and knowledge of how the tasks are developed, who is in charge, and so on, the tool have a user interface that show the progress of each task.


Conference workflow

Task	Resource	State
Find the program committee	Chloé Lambin	No started
Prepare the call for paper	Jacques Khelil	No started
Install conference tool	Ellen Martin	No started
Assign papers to reviewers	Angela Buker	No started
Submit paper	A-1, A-2, A-3	No started

Figure 18: Task progress list

6. Conclusion and future work

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).

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.

Acknowledgments We gratefully thank the support from the SIMILAR network of excellence, supported by the 6th Framework Program of the European Commission, under contract FP6-IST1-2003-507609 (http://www.similar.cc), the CONACYT (www.conacyt.mx) program supported by the Mexican government.

7. References

[1] Annett, J., Cunningham, D., & Mathias-Jones, P. (2000). A method for measuring team skills. *Ergonomics, 43*, 1076–1094.

[2] Annett, J., & Duncan, K. (1967). Task analysis and training design. *Occupational Psychology, 41*, 211–227.

[3] Beard, D. V., Smith, D. K., & Denelsbeck, K. M. (1996). QGOMS: A direct-manipulation tool for simple GOMS models. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI '96)* (Vol. 2; pp. 25–26). New York: ACM Press.

[4] Barthet, M.-F., & Tarby, J.-C. (1996). *The Diane+ method*. In J. Vanderdonckt, (Ed.), *Computer-aided design of user interfaces* (pp. 95–120). Namur, Belgium: Presses Universitaires de Namur.

[5] Baumeister, L. K., John, B. E., & Byrne, M. D. (2000). A comparison of tools for building GOMS models: Tools for design. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2000)* (Vol. 1; pp. 502–509). New York: ACM Press.

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

[7] 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.

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

[9] 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).

[10] 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, to appear. Available at: http://wwwcs.uni-paderborn.de/cs/ag-szwillus/lehre/ws05_06/PG/PGAMBOSS/index.php

[11] 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).

[12] 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).

[13] John, B. E.,&Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction, 3*, 320–351.

[14] 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).

[15] Mahfoudhi, A., Abed, M.,&Tabary,D. (2001). From the formal specifications of user tasks to the automatic generation of the HCI specifications. In A. Blandford, J. Vanderdonckt, & P. Gray, (Eds.), *People and computers XV* (pp. 331–347). London: Springer.

[16] 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).

[17] 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.

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

[19] Pinelle, D., Gutwing, C., and Greenberg, S. Task Analysis for Groupware Usability Evaluation: Modeling Shared-Workspace Tasks with the Mechanics of Collaboration. ACM Transactions on Computer-Human Interaction10, 4 (2003), 281-311.

[20] 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).

[21] 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).

[22] 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).

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

[24] 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

[25] 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.

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