Conceptual frameworks for the development of CSCW systems

Cléver Ricardo Guareis de Farias, Luís Ferreira Pires, Marten van Sinderen

Centre for Telematics and Information Technology, University of Twente
PO Box 217, 7500 AE, Enschede, The Netherlands
{farias, pires, sinderen}@cs.utwente.nl

Abstract: Models and theories concerning cooperation have long been recognised as an important aid in the development of Computer Supported Cooperative Work (CSCW) systems. However, there is no consensus regarding the set of concepts and abstractions that should underlie such models and theories. Furthermore, common patterns are hard to discern in different models and theories. This paper analyses a number of existing models and theories, and proposes a generic conceptual framework based on the strengths and commonalities of these models. We analyse five different developments, viz., Coordination Theory, Activity Theory, Task Manager model, Action/Interaction Theory and Object-Oriented Activity Support model, to propose a generic model based on four key concepts common to these developments, viz. activity, actor, information and service.

1. Introduction

The development of Computer Supported Cooperative Work (CSCW) systems is not a trivial task, since it involves problems originated in two distinct areas, viz. social sciences and distribute systems. Interdisciplinary issues, such as the articulation support of cooperative procedures, the sharing of information space and adaptation of new technologies to the organisation, have long been identified as core requirements that must be addressed during the development of cooperative systems [1].

Three alternative approaches are normally used in the development of cooperative systems: ad hoc development, use of CSCW toolkits and use of integrated CSCW environments. In the ad hoc development the system is usually built from scratch, making little or no use of elaborate components. CSCW toolkits, such as Prospero [2], DistEdit [10] and Groupkit [17], allows one to build a system based on a set of pre-defined building blocks that can be reused and combined in different ways. Integrated CSCW environments, such as the activity support system TACTS [20] and the locale support system Worlds [6,7], are cooperative platforms in which a set of abstractions and facilities is provided to the development of CSCW systems.

Cooperative models and theories have been successfully used in the development of CSCW systems. These models and theories represent application domain knowledge and constitute in a conceptual framework that captures some of the most important aspects concerning the computer support of a cooperative activity. Models and theories guide the developers of cooperative systems, by helping them focusing on the most relevant issues as soon as possible, allowing them to structure the system in a coherent way.

Currently, we are investigating an integrated software development process for the design of cooperative systems. In the beginning of this investigation we had to choose a model for the

initial phases of the development process. We observed in the literature that multiple models and theories for modelling cooperative applications have been developed, such that we decided to study and compose these models to obtain enough information for developing our own model. This activity has been inspired by the divergence between the available CSCW models and our initial assumption that we could develop a model that profit from many individual benefits of these models.

This paper aims at analysing a set of CSCW models and theories and proposing a generic conceptual framework based on the strengths and commonalities of these models and theories. We analyse five different models and theories, viz., Coordination Theory [14], Activity Theory [12], Task Manager [11], Action/Interaction Theory [6] and Object-Oriented Activity Support model [20]. Being a tool, Task Manager seems a bit odd in this list, but this poses no problem since we are only interested in the cooperative model employed in the development of Task Manager.

To perform a fair analysis on a number of different models and theories and to produce generally useful results we use a systematic approach based on well-defined modelling techniques. We use the Unified Modeling Language (UML) [16] to capture and analyse the main concepts present in each cooperative model or theory. UML is an OMG's standard language for modelling software systems artefacts.

An alternative approach to produce a generic conceptual model consists of performing domain analysis on several CSCW applications, such as shared whiteboards, chats, workflow management systems and co-authoring systems, in order to identify the concepts common to most of these systems. This approach would be particularly suitable if we were interested in capturing the concepts that concerns to a specific domain. An example of this approach can be found in [19], where a conceptual model of a workflow process is presented.

The remaining of this document is organised as follows. Section 2 outlines the approach used to capture the concepts present in the different models. Section 3 reports a number of CSCW conceptual models and theories and their corresponding UML representation. Section 4 presents an analysis of these developments and introduces our conceptual cooperative work model, while section 5 compares our conceptual model with these developments and related approaches. Finally, section 6 draws some conclusions and gives some ideas for future work.

2. Approach

In order to systematically capture and analyse the concepts underlying a number of cooperative models and theories, we adopted the OMG standard modelling language UML [16]. For each cooperative model or tool described in the next section, we develop object models that help understanding its main concepts. The development of an object model aims at capturing the concepts present in the application domain under investigation. At this abstraction level, there is not necessarily a direct mapping between the identified concepts and an implementation of the model. Such a high level description of the system is commonly called *conceptual perspective* in the literature [8].

An *object* is a concept, an abstraction or thing with crisp boundaries and meaning for the problem at hand [18]. An *object class* is a representation of a group of objects with similar structure (attributes), common behaviour (operations), common relationships to other objects (links), and common semantics.

There are two types of object models, viz. *class diagram* and *instance diagram*. A class diagram represents object classes and relationships (associations) among these classes. An instance diagram represents object and links among these objects. A class diagram describes a potentially infinite number of instance diagrams, while an instance diagram is one particular instance of a class diagram.

The steps towards the representation of a CSCW model or tool are the following:

- identification of object classes. An object in a particular model can represent either a concrete entity, such as an actor, a document or a tool, or an abstract entity, such as an activity or a task;
- identification of binary associations between classes. An association represents a group of links between two objects, such that these links have a common structure and a common semantics. Any dependency between two or more classes is represented as an association. In principle associations can be binary, ternary or n-ary. However, for simplification purposes, dependencies between objects of the same class are considered as binary associations. The multiplicity of an association, i.e., how many instances of one class may relate to a single instance of an associated class, is also specified during this step;
- identification of association classes. An association class is an association with class properties, i.e., an association that can have attributes, operations and other features;
- identification of the main attributes. Classes may have attributes. However, the identification of attributes is not mandatory in our approach. Attributes are identified whenever they are clearly stated in the description of the CSCW model and are important to the comprehension of the model;
- instantiating the class diagram. Examples are used to instantiate the class diagram to check the consistency of the model.

Because we are only interested in the conceptual perspective of the system, we do not consider operations in our models.

3. Conceptual frameworks for CSCW

This section presents a number of theoretical models, theories and underlying systems that have been proposed to the design of CSCW applications. For each model we present a brief informal description as found in the literature, its object representation and the main modelling decisions we have taken according to our interpretation of the model. The object representation of a model consists of one or more class diagrams; instance diagrams are omitted in this paper for conciseness.

3.1. Coordination theory

Coordination theory consists of a set of principles that allows one to manage interdependencies between activities performed to achieve a goal [14]. This theory concentrates on coordination problems. Typical examples of coordination problems are the identification of goals, the mapping of goals to activities, the ordering of activities, the selection of actors to perform an activity, the management of interdependencies between activities and the allocation of resources for an activity.

Several disciplines provide different and complementary insights for understanding coordination. A conceptual framework to better understand and analyse coordination issues has been proposed in [14]. This framework consists of some concepts, viz. goals, activities, actors, resources and interdependencies.

One or more actors perform one or more activities, which are determined by some goals. Activities depend on each other and their interdependence can be characterised in terms of objects common to the activities. These common objects, such as shared resources, constrain how each activity is performed. Three generic kinds of interdependence are possible: prerequisite, i.e., the output from an activity is required by another, common resource, i.e., a resource is required by multiple activities, and simultaneity, i.e., two or more activities must occur at the same time.

Figure 1 presents our class diagram for the Coordination theory.

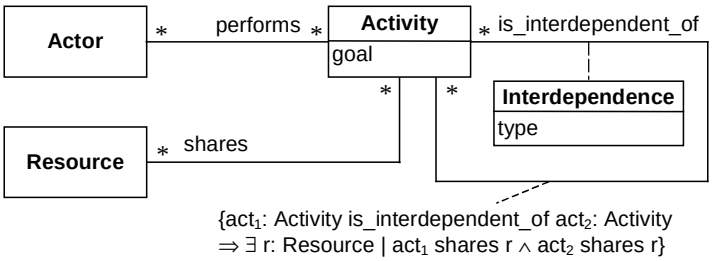


Figure 1. Coordination theory class diagram.

The identification of the key concepts of the Coordination theory is straightforward. The concepts of actor, activity and resource are mapped onto the object classes Actor, Activity and Resource, respectively. The concept of goal is mapped onto a class attribute, while the concept of interdependence is mapped onto an association class. In the associations between two classes represented in a class diagram, the multiplicity symbol '∗' represents zero or more instances of a class may be contained in a given association at a time.

A goal is represented as an attribute of an activity because a goal is only meaningful in the context of an activity and does not depend on other concepts. Similarly, the type of interdependence is represented as an attribute because it can be seen as a property of the association between two activities. Furthermore, one activity can only be interdependent of another if there is a resource that is shared by both activities (see constrain in Figure 1).

3.2. Activity theory

Activity theory [12] focuses on (1) the use of the activity concept as the basic unit of analysis, (2) the development of an activity according to its past and (3) the mediation of the relations within activities.

The Activity theory considers an activity as being a collective phenomenon that transforms a material object. The objective of an activity is accomplished through the transformation of this object towards some desired state.

An activity involves a community of participants. However, only part of this community understands the purpose of the activity, the so-called active participants. The active participants, also called active subjects, can be either individual or collective. The participants of an activity perform this activity through conscious and purposeful actions.

The cultural mediation of the relationships within an activity is also considered in this theory. Tools mediate the relationship between the active subject and the material object, while rules mediate the relationship between the active subject and the others participants, and division of labour mediates the relationship between the object and the community.

Figure 2 depicts our class diagram for the Activity Theory.

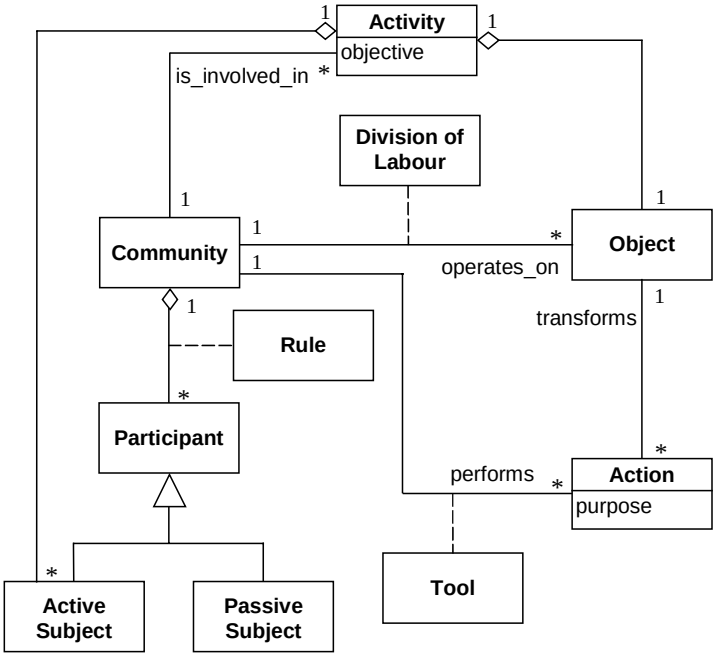


Figure 2. Activity Theory class diagram.

The identification of the main concepts is not as trivial as in the case of the Coordination theory. The concepts of community, active subject, material object and activity have been identified in a first step and modelled as object classes Community, Passive Subject, Object and Activity, respectively. The object class Passive Subject has been defined to represent all the participants involved in an activity that are not active subjects. The object classes Active Subject and Passive Subject have been generalised into the class Participant. Each instance of the class Participant is part of a Community. In the associations between two classes represented in a class diagram, the multiplicity symbol '1' represents that exactly one instance of a class participates in a given association at a time.

While analysing the Activity theory we noticed that the relationship between the active subject and the activity was stronger than the relationship between the community and the activity. In this way, we model an Active Subject as part of an Activity, i.e., as an aggregation association, while a Community is only associated with an Activity. An aggregation association is graphically represented by a hollow diamond. The community operates on the material object and realises an activity performing a set of purposeful actions (class Action), which transform the material object.

Rules regulate the relationship between the community and the active subject and, therefore, are modelled as the association class Rule, which is established between the classes Community and Participant. The division of labour mediates the relationship between a community and a material object. Thus, the division of labour is also modelled as an association class, called Division of Labour, which is established between the classes Community and Object. Tools are the instruments used by a community to perform an action. Hence, tools are mod-

elled as the association class Tool, which is established between the classes Community and Action.

3.3. Task Manager

The Task Manager [11] is a tool developed for the specification and management of cooperative work. Task Manager has been developed around the concept of task. People perform a common task by making use of shared documents or services, and communicate with each other by exchanging messages.

A task can be seen from different points of view: (1) as a project, (2) as a set of subtasks with the establishment of dependencies between them or (3) as a folder, used only as a container of shared objects, such as subtasks, documents and messages.

The resources needed to achieve the goal of a task can be either "pointers" to several computerised objects, such as documents, or simple objects, such as rooms, budgets and machinery. The latter kind of resources is handled by services implemented outside the scope of Task Manager.

People can have different levels of participation in a task. Some people, called participants, have equal access rights to the attributes of a task, its documents and services and its messages, while other people, called observers, are only interested in the completion of a task. The latter only have read access to the information related to the task. A special participant who is responsible for the execution of the task has more access rights than the other participants. All the people involved in a task can exchange electronic mail messages within the context of the task.

Figure 3 shows our class diagram for Task Manager.

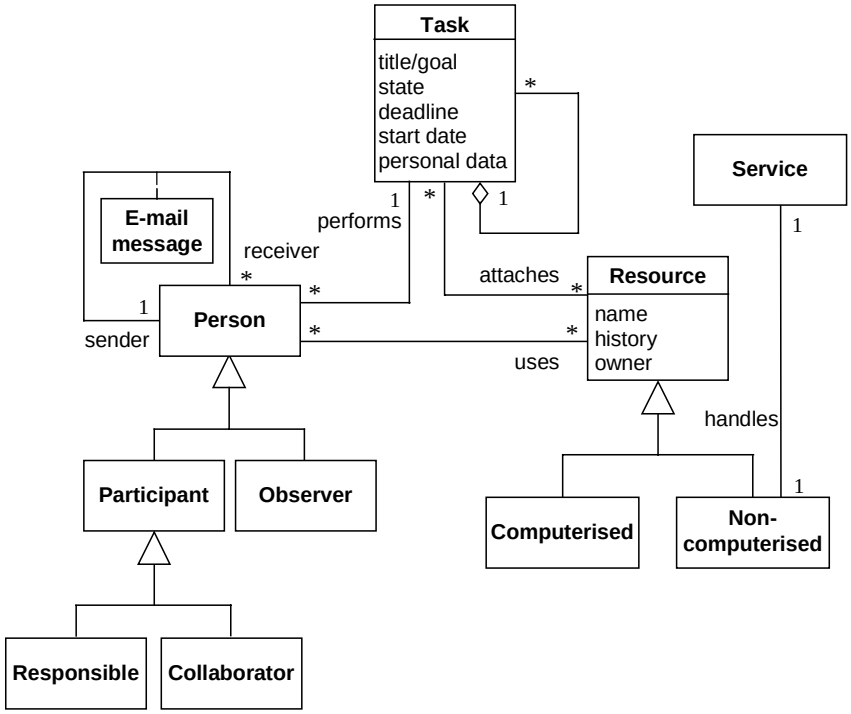


Figure 3. Task Manager class diagram.

Within Task Manager, the main concepts identified are tasks, people and resources, which are represented by the object classes Task, Person and Resource, respectively. A task may be

decomposed into multiple subtasks; this is represented by the aggregation association between the class Task and itself.

The class Person has been specialised into the classes Participant and Observer. The class Participant has been specialised into the classes Responsible and Collaborator. These classes represent all possible participation levels in a task. Messages are represented as an association class between persons with sender and receiver roles, respectively.

The class Resource has been specialised into the classes Computerised and Non-computerised resources. Non-computerised resources are handled by external services, represented by the class Service.

3.4. Action/Interaction theory

The theory of action and interaction has been the basis for the development of a CSCW environment called WORLDS [6,7]. In particular, two aspects of this theory have been emphasised in the development of WORLDS, viz. the notions of action and interaction and the notion of social world.

The concepts of actions and interactions are the main concepts in this theory. Actions are embedded within courses of interaction and occur within the context of dynamic structural conditions. Structural conditions may either restrain or facilitate courses of interactions. Courses of interaction are composed by sequences of connected actions and they can be either static or dynamic. A trajectory is a course of actions over time, and the actions and interactions that contribute to the evolution of the trajectory.

Actions are carried out by interactants. An interactant controls a trajectory and its course of interactions through their actions. An interactant can be either individual or collective. Articulation or alignment of actions is necessary when multiple interactants are involved.

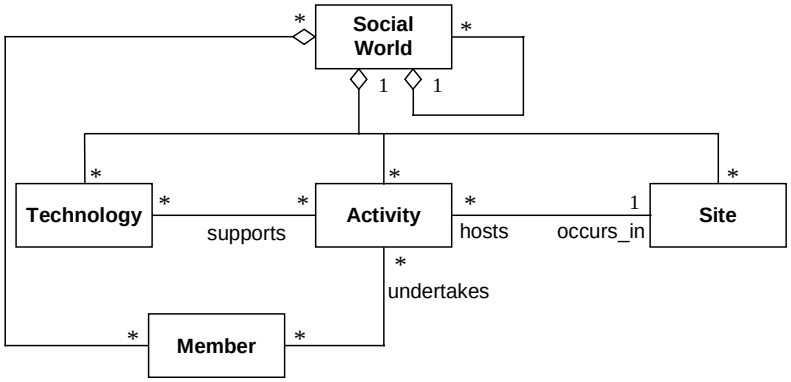
Social worlds are interactive units that arise when a group of individuals acts in a collective way. A social world has activities, sites where activities occur and technology as means to carry out the activities. A social world can also have sub-worlds and individuals can be members of several social worlds simultaneously. The work undertaken by the members of a social world can be described using the action/interaction notions.

Figure 4 shows our class diagram for the Action/Interaction theory.

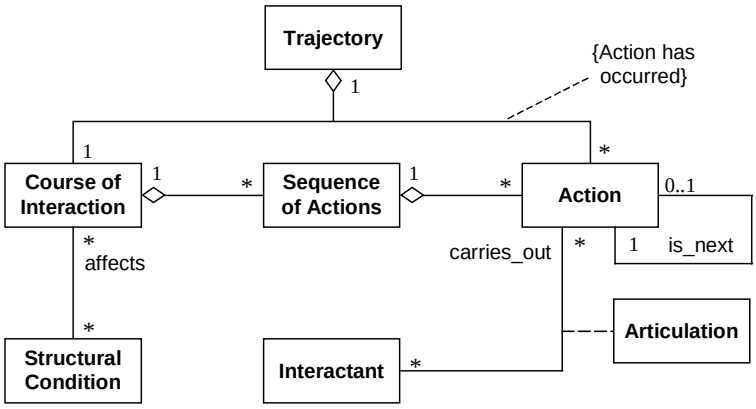
The modelling of the Action/Interaction theory using a class diagram has been a difficult task. Due to the conciseness of the presentation of the main concepts of this theory in the literature, we found it difficult to understand and precisely represent some relationships between concepts. To avoid introducing relations between concepts that have not originally meant in this theory, we decided to split the object representation of the theory into two classes diagrams, viz., one describing the action/interaction concepts (see Figure 4(a)) and another representing the social world concepts (see Figure 4(b)).

Concerning social world we identified the following concepts: social world, activity, technology, site and member. Each one of these concepts is represented as an object class, viz., Social World, Activity, Technology, Site and Member, respectively.

Concerning action/interaction we identified the following concepts: trajectory, course of interaction, structural condition, sequence of actions, action and interactant. Each one of these concepts was mapped onto a corresponding object class, viz., Trajectory, Course of Interaction, Structural Condition, Sequence of Actions, Action and Interactant, respectively.



(a) Social world concepts



(b) Action/interaction concepts

Figure 4. Action/Interaction theory class diagram.

A Trajectory consists of a Course of Interaction and a set of Actions. However, only the actions that have been performed belong to a trajectory. This property is represented by one constrain in the association between Trajectory and Action. A Sequence of Actions consists of a set of Action objects connected to each other. The association is_next represents the connection between two Action objects. This association implies that each Action may be associated with at most one other Action, which is represented by the '0..1' multiplicity in the class diagram.

Another important feature of this model is the need for articulation or alignment of actions when multiple interactants are involved. Because an action can be performed by several interactants, articulation is necessary among the actions that can be performed by a certain interactant. This requirement implied in the definition of the association class Articulation between the object classes Interactant and Action.

3.5. Object-Oriented Activity Support

The Object-Oriented Activity Support Model (OOActSM) [20] aims at providing an integrated framework for CSCW systems.

The concept of activity is the basic abstract concept of OOActSM. An activity is a structured object possibly containing an arbitrary number of sub-activities. Activities are executed by an actor, which can be a single person, a group of persons or a programmed autonomous agent in a computer. Activities also have a context, representing elements created or manipulated by the activity (e.g., documents), elements used to perform an activity (e.g., tools or documents), other people involved in the activity (e.g., an organisation or a department) and information items required by the activity (e.g., goals or policies).

The OOActSM allows one to represent an activity or sub-activity, its actor and the objects in the context of the activity as object classes. The objects in the context of an activity can have a context themselves. Connections between all the elements within this model are established using object association and object references.

An activity has a state, an execution history and an execution procedure. The state and the operations of an activity determine its behaviour.

Figure 5 depicts our class diagram for OOActSM.

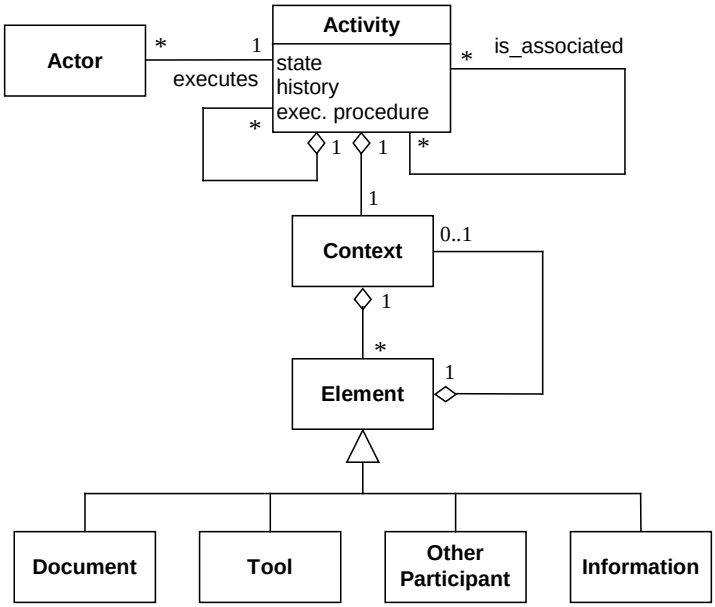


Figure 5. OOActSM class diagram.

During the analysis of OOActSM we identified the concepts of activity, actor, context, document, tool, other participant and information, which are modelled as the object classes Activity, Actor, Context, Document, Tool, Other Participant and Information, respectively.

An Activity object can aggregate many Activity objects, each one representing a sub-activity. Additionally, an Activity can be associated with multiple Activity objects through the association is_associated. Each Activity has one Context, which can include elements such as Document, Tool, Other Participant and Information. Each element in the context of an activity may have its own context. To fulfil these requirements we define the object class Element, which generalises all the object classes that can be in the context of an activity; we also define that instances of Element are contained in an instance of object class Context.

4. A generic model for cooperative work

This section analyses the CSCW models described previously and proposes a new model based on the strengths and similarities of these models.

4.1. Analysis of CSCW models

In the foregoing models, we can observe that four concepts are usually present in most of these models (although sometimes with different names): activity, actor, resource and tool. Table 1 summarises the terminology used by the different models.

Table 1. Terminology comparison.

	Activity	**Actor**	**Resource**	**Tool**
Coordination theory	activity	actor	resource	-----
Activity theory	activity/action	participant, subject	object	tool
Task Manager	task/subtask	person, participant, observer	resource	-----
Action/Interaction theory	activity/action	member, interactant	-----	technology
OOActSM	activity/sub-activity	actor	document, information	tool

An activity, also called task in Task Manager, represents a cooperative procedure. An activity can usually be decomposed into smaller parts, called sub-activities, subtasks or actions, and can usually be related with other activities.

An actor, also called subject, person, member or interactant, represents an entity responsible for performing an activity. An actor can be either a human being or an autonomous agent and can also be either individual or collective.

A resource, also called object, document or information, represents something that is used, produced or transformed by an activity. A resource can be available in an electronic form or in any other form.

A tool, also called technology or artefact, is an entity that provides the computer support for the execution of an activity. A tool can be either a specific groupware system, such as a co-authoring editor or a videoconferencing system, or a non-groupware system, such as a text processor, an electronic mail system or a database management system.

The characteristics of a concept sometimes differ from one model to the other. The differences are mainly due to the different abstraction levels at which the concept is considered. For example:

- an activity can be decomposed into smaller units according to Activity theory, Task Manager, Action/interaction theory and OOActSM. However, an activity is viewed as an atomic concept in the Coordination theory;
- no distinction is made between actors in the Coordination theory, Action/Interaction theory and OOActSM. However, different kinds of actors are possible in the Activity theory and in the Task Manager. Although the Action/Interaction theory does not consider different kinds of actors, the realisation of this theory into the WORLDS prototype considers different kinds of actors. Both Activity Theory and Task Manager consider different kinds of actors, but in these models actors are associated with a single activity;
- only the Task Manager model considers different kinds of resources (computerised and non-computerised resources).

Besides the possible different names and characteristics of a concept in each model, a concept can also differ in the way it relates with the other concepts. For example:

- only the Activity theory considers an actor as being part of an activity. In the other models an actor is associated with an activity with the intention of performing this activity;

- a resource is shared by activities according to the Coordination theory and Task Manager, while according to Activity theory and OOActSM a resource is part of an activity;
- a resource is used by an actor in the Activity theory and Task Manager, while the Coordination theory and OOActSM do not consider this relationship, and;
- the role of tools also varies considerably. Tools are not considered in the Coordination theory and in the Task Manager. A tool is used by an actor to perform an action that transforms a resource in the Activity theory. A tool supports an activity in the Action/interaction theory. A tool is part of an activity in the OOActSM;
- the communication between the different actors involved in an activity is only explicitly represented in the Task Manager model.

4.2. Our conceptual model

Above we identified many similarities and differences among CSCW models. However, none of these models seems to be general and complete enough. Some of the relationships, such as actors using resources and tools to perform a task, are obvious and do not seem to be relevant to the understanding and modelling of CSCW systems. Therefore, we decided to propose a new model that exploits the similarities encountered in the analysis and profits from the strengths of each particular model.

Figure 6 depicts our generic conceptual CSCW framework. This framework is based on four concepts, viz. activity, actor, information and service, and on a set of relationships between them, with emphasis on the relationships between an activity and the other concepts. These concepts keep a close relationship with the main concepts identified in the previous section, viz. activity, actor, resource and tool, respectively.

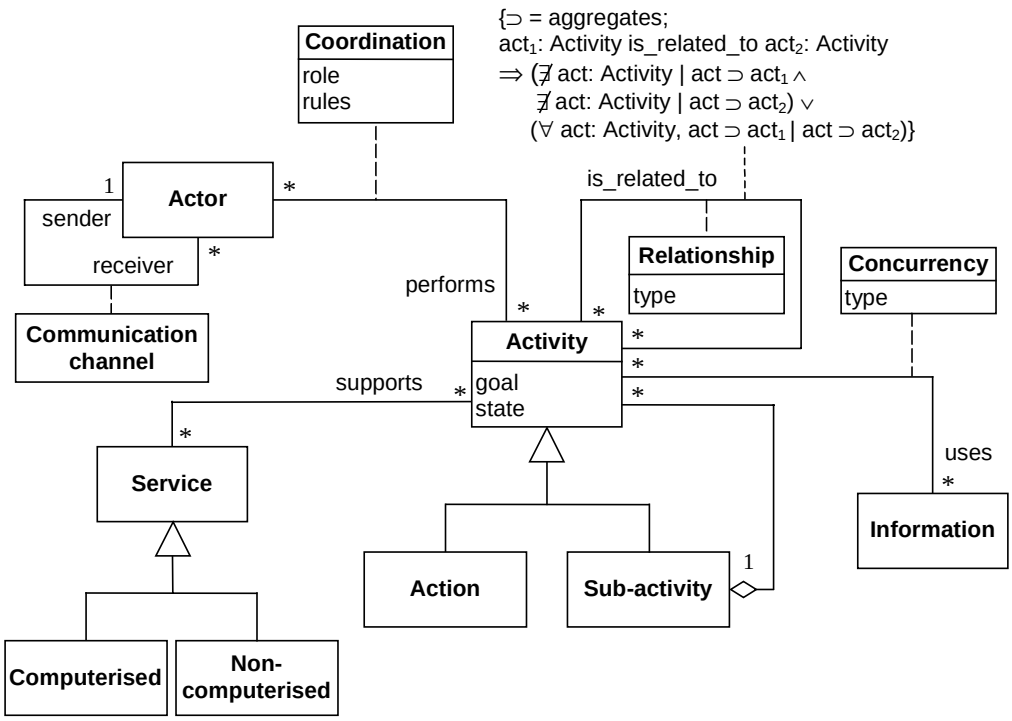


Figure 6. A generic CSCW model.

An activity is the basic unit of analysis and design, and represents a cooperative procedure. Activities have at least two properties, viz. a goal and a state. An activity can be decomposed into sub-activities and actions. The differences between a sub-activity and an action are two-fold: (1) a sub-activity can be further decomposed while an action is atomic, i.e., it can not be further decomposed at that abstraction level; and (2) a sub-activity is considered long-lived, i.e., an activity can be performed in minutes, hours, days and even months, while an action abstracts from duration, by only considering the moment when the action is finished.

A relationship association can be established between two activities. However, this association is restricted to activities at the same aggregation level (see constrain in Figure 6). One activity may relate to another activity if these activities are top-level activities, i.e., they are not part of other activities, or these activities are part of the same activity. Examples of relationship types include, amongst others, the disabling of an activity by another, the sequential execution of two activities, the synchronised execution of two activities and so forth. Activities execute concurrently, unless they are constrained by, for example, a sequential relationship.

An actor is an active entity, which is responsible for performing an activity. Actors must communicate with each other to perform the activity properly. In Task Manager, this communication is represented by the exchange of e-mail messages. In our model, we generalise this idea by stating that a Communication channel must be present between the actors of an activity. The communication channel may represent an audio channel, such as a telephone line, a videoconferencing channel, an electronic mail tool or even a live channel, in which the actors are colocated at the same room. In this way, communication messages are exchanged between two or more actors, an actor playing a sender role and possibly multiple actors playing a receiver role.

An actor is not part of an activity but is associated with one or more activities. A role and a set of coordination rules are the attributes of the Coordination association class established between workers and activities. A role is used to describe the part taken by an actor in an association. In this way, an actor involved in several activities can play different roles in each one of the activities. Coordination rules, such as policies and floor control mechanisms, regulate the relationship between different actors performing the same activity.

A piece of information represents any kind of electronic data used by an activity, such as messages, documents or database records. Information can either be consumed or produced by an activity. The concept of information corresponds to the concept of resource identified in the previous section. However, our model does not distinguish between computerised and non-computerised information. We assume that at least an electronic reference to a non-computerised piece of information should be available to enable the automated cooperative work activity to keep track of this information. This reference can have the form of a name or some number (identifier).

Frequently, the same information is shared by multiple activities. To cope with the simultaneous access to a piece of shared information we identify two alternatives: locking mechanisms in the information itself or concurrency control mechanisms to monitor the access to the information. The former alternative is rather simple, but at the same time limited. The latter alternative is used to provide more elaborate mechanisms, such as non-locking shared write access and interactive concurrency control [3], often required by cooperative systems. Concurrency control mechanisms are provided by the Concurrency association class established between the activity and the shared information.

Finally, the service concept represents any kind of computerised or non-computerised service that supports the execution of an activity. A service comprehends a tool, as characterised in the previous section, and extends this concept to non-computerised services, such as the service provided by a secretary while typing a letter or the service provided by a room in holding a conference.

5. Discussion

This section compares our modelling approach with the models reported in this paper and with other related approaches.

5.1. Presented models

The Coordination theory has three main limitations with respect to our model: (1) it does not consider the role of tools, (2) it does not consider the structuring of an activity into smaller units, and (3) although the Coordination theory allows the establishment of interdependence between two activities, our model extends this concept of interdependence to a wide variety of relationships.

The main problem with the Activity theory is that it does not allow the representation of an activity as a collection of smaller units. One can assume that an activity can be decomposed into a set of actions. Nevertheless, the relationship between an action and an activity is not clearly defined in this theory. Besides, the relationship between two or more activities/actions is not considered. Our model, in contrast, provides means to precisely define an activity and its related sub-components, viz., action and sub-activity. Furthermore, the relationship between these concepts can also be clearly defined using our model.

The Division of Labour concept in the Activity theory is comparable to the Coordination concept in our model. However, our coordination concept offers more capabilities, such as policies and floor control mechanisms.

There are three shortcomings in the Task Manager model: (1) the absence of relationships between two or more tasks, except for the parent/children relationship, (2) the absence of a concept representing tools in the model, which is a consequence of the model being implemented by a tool itself, and (3) in the description of the model there are no evidences that the same person can perform several tasks, although it seems obvious that this situation should be possible. However, a fixed role for each person can be considered as a drawback of this model because it limits its flexibility.

The Action/Interaction theory also has a few limitations compared to our model. The first limitation is the absence of an explicit representation for the resources used by an activity. We can implicitly assume that a site holds these resources in this model, but we encourage the explicit representation of such information.

The effectiveness of the Action/Interaction model is also restricted by the type of relationship between two actions considered in this model. The relationship between two actions is rather simple and rigid if compared to our model. The Action/Interaction theory has only the is_next relationship, which enables actions to be organised into sequences of actions. Nevertheless, our model provides an equivalent type of relationship, viz., the enabling relationship, and other types of relationships, such as disabling, choice, synchronisation and so on. Furthermore, besides enabling the specification of the relationship between the sub-elements that

compose an activity, our model allows the establishment of relationships between related activities, which is not supported by the Action/Interaction theory.

The role of technology in the Action/Interaction theory is close to the role of the service concept in our model. However, the former takes into account only computerised tools or services, while our model considers both computerised and non-computerised services.

The OOActSM presents a number of similarities with our model, however we can still identify a few problems in this model. First, the OOActSM model allows the definition of relationships between two activities, but the model does not mention which types of relationships can be defined. Second, there is no clear distinction between the resources required by an activity and the tools needed to support this activity, in the sense that both (resources and tools) belong to the context of the activity. Finally, we doubt the usefulness of the concepts of tools, other participants and resources in the model, because these concepts are described as having the same kind of relationship with respect to an activity, i.e., they all belong to the Context of an Activity.

Although concurrency control is an important issue in distributed systems in general, most of the models presented so far ignore this issue. Our model explicitly takes this issue into account, by supporting the representation of concurrent access to shared information.

5.2. Other related models

Another interesting approach to describe and compare cooperative systems is the conceptual model presented in [4]. According to this model, a cooperative system consists of three different and complementary models, viz., ontological, coordination and interface.

The ontological model is a static description of the objects and operations that the system provides to the user, while the interface model is a description of how the users interact with the system and with each other. The coordination model is a dynamic description of the activities each participant may perform and how these activities are coordinated. This model is more relevant than the other models in the context of our research.

An activity is the main concept in the coordination model. An activity is described as a set of operations with a defined goal that an actor playing a certain role can perform. An operation is a transformation performed on an object. An actor may be a single user, a group of users or a computer system.

A procedure is defined as a set of activities and the ordering among them. An endeavour consists of instances of activities. An instance of an activity, called task, can be inactive or active. An inactive task has not yet started or has already been completed, while an active task has already started but has not been completed. An activity is initiated by a start action and completed by a termination action. Typically, the actor who performs an activity also executes one or all of its start and termination actions.

There are two levels of coordination in this model, viz., the activity level and the object level. The activity-level coordination describes the sequencing of activities that constitutes a procedure, while the object-level coordination describes how sequential or simultaneous access to the same objects can be managed.

The coordination model also provides four levels of concurrency, viz., sequential, parallel, additive concurrent and fully concurrent. These types of concurrency have been identified based on the simultaneity of the activities and how these activities access a shared resource.

Comparing this conceptual model, and in particular its coordination model to the model we propose in this paper, one can conclude that these models are similar. Most of the concepts present in the coordination model can be found in our approach, sometimes with the same or similar semantics. The main differences are the following:

- in the coordination model there is no support for the structuring of an activity into smaller units. Operations can be considered as sub-units of an activity, but still one can not decompose operations further. Our model supports the structuring of an activity into sub-activities and actions;
- actions in the coordination model are not part of an activity. Although actions can be performed by actors in the same way as they perform an activity (most of the time by the same actors);
- our model does not make a distinction between activity-level and object-level coordination, although it provides the same capabilities: the activity-level coordination model is similar to the description of the activities and their relationship in our model, while the object-level coordination model is similar to the description of the relationship between the activities and the set of information used by these activities in our model.

The major differences between this conceptual model and our model can be noticed in its interface model. Because our model is focused on the design of the activities supported by a cooperative system from the user point of view, and information and services that are necessary to support and carry out these activities, little attention is given in our model to issues related to user interface.

Considering the interface model, only the concept of context can be found back in our model, although with subtle differences. The role of tools is relegated to a second level of importance. For example: in the context of an activity we can find a reference to a tool which might be used by the activity. In our model the services that support an activity are almost as important as the activity itself. Besides, we take into account both computerised and non-computerised services.

Some other CSCW systems are based on concepts that are at a different conceptual level than the theories and models discussed. These systems are used to build cooperative systems and are often called meta-groupware [4]. Examples of meta-groupware include Oval [15] and Lotus Notes [13]. Because the concepts present in meta-groupware systems are generative concepts, in this paper we have decided to ignore these systems in our research for the time being.

6. Conclusion

This paper analyses a number of cooperative models and theories. The main purpose of the work is to identify and represent their main concepts and abstractions used by of these models and theories.

The analysis performed revealed that the different models and theories actually have a set of common concepts that do not differ much in each of them. This implies that a common-sense model can be developed by considering the similarities and strengths of these models and theories.

Four concepts have been identified in this paper: activity, actor, service and information. The paper shows that these concepts are generic, since they appear in similar forms in almost all models studied.

The concept of activity and, particularly, the concepts of sub-activity and action provide a convenient framework to represent a collaborative activity at different abstraction levels. In our model, actors perform an activity according to a certain role. Coordination rules regulate the relationship between different actors performing the same activity. Actors also communicate with each other while performing an activity. An activity operates in a shared set of information producing, consuming or changing it. Concurrency control mechanisms should be use to regulate the access to shared piece of information. A set of external services collaborates in the support of the cooperative activity. We made no assumptions with respect to the allowed type of services. Both computerised and non-computerised services are possible.

The concepts proposed in this work have been successfully used in the modelling of some cooperative applications, such as a travel claim system, a paper reviewing application and a brainstorming conferencing application. These examples are reported in appendixes A, B and C, respectively.

The work presented here is only the first step towards the development of what we call a comprehensive framework for building cooperative systems. Currently we are working on the development of a combined approach for developing cooperative components. In the future we intend to formalise the concepts presented here, by extending the metamodel of UML to include these concepts. The idea is to integrate the concepts in the development process. We also intend to apply this comprehensive framework in the development of a telelearning environment consisting of several cooperative applications.

Acknowledgements

This work has been carried out in the scope of the Amidst (Application of Middleware in Services for Telematics) project, which is a project of the 'Telematica Instituut' (the Netherlands). Cléver Ricardo Guareis de Farias is supported by CNPq (Brazil).

References

1. Bannon, L.J. and Schmidt, K.: CSCW: Four Characters in Search of a Context. *Proceedings of First European Conference on Computer Supported Cooperative Work (ECSCW'89)*, pp. 358-372, 1989.
2. Dourish, P.: Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications. *ACM Transactions on Computer-Human Interaction*, Vol. 5, Nº 2, pp. 109-155, 1998.
3. Ellis, C.A., Gibbs, S.J. and Rein, G.L.: Groupware: Some issues and experiences. *Communications of the ACM*, 34 (1), pp. 38-58, 1991.
4. Ellis, C. and Wainer, J.: A conceptual Model of Groupware. *Proceedings of the of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, pp. 79-88, 1994.
5. Farias, C. R. G., Pires, L. F., and Sinderen, M. van: Conceptual frameworks for the development of CSCW systems. *Technical Report CTIT 99-07*, University of Twente, 1999.

6. Fitzpatrick, G., Tolone, W.J. and Kaplan, S.: M. Work, Locales and Distributed Social Worlds. *Proceedings of the 1995 European Conference on Computer Supported Cooperative Work (ECSCW '95)*, pp. 1-16, 1995.

7. Fitzpatrick, G., Kaplan, S. and Mansfield, T.: Physical Spaces, Virtual Places and Social Worlds: A Study of Work in the Virtual. *Proceedings of the ACM 1996 Conference on Computer Supported Work (CSCW'96)*, pp. 334-343, 1996.

8. Fowler, M. and Scott, K.: *Uml Distilled: Applying the Standard Object Modeling Language*. MA: Addison-Wesley, 1997.

9. Jablonski, S.: MOBILE: A Modular Workflow Model and Architecture. *Proceedings of the International Working Conference on Dynamic Modelling and Information Systems*, pp. 1-30, 1994.

10. Knister, M.J. and Prakash, A.: DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors. *Proceedings of the ACM 1990 Conference on Computer-Supported Cooperative Work (CSCW'90)*, pp. 343-355, 1990.

11. Kreifelts, T., Hinrichs, E. and Woetzel, G.: Sharing To-Do Lists with a Distributed Task Manager. *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*, pp. 31-46, 1993.

12. Kuutti, K.: The concept of activity as a basic unit of analysis for CSCW research. *Proceedings of the Second European Conference on Computer Supported Cooperative Work (ECSCW'91)*, pp. 249-264, 1991.

13. Lotus Corporation. Lotus Notes/Domino. Internet: http://www.lotus.com.

14. Malone, T. W. and Crowston, K.: What is Coordination Theory and how can it help design cooperative work systems? *Proceedings of the of the 1990 ACM Conference on Computer Supported Cooperative Work (CSCW'90)*, pp. 357-370, 1990.

15. Malone, T. W., Lai, K.Y. and Fry, C.: Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. *Proceedings of the of the 1992 ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, pp. 289-297, 1992.

16. Object Management Group: *OMG Unified Modeling Language Specification*, 1997. Internet: http://www.omg.org.

17. Roseman, M. and Greenberg, S.: Building Real Time Groupware with GroupKit, A Groupware Toolkit. *ACM Transactions on Computer Human Interaction*, 3(1), pp. 66-106, 1996.

18. Rumbaugh, J., Blaham M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-oriented modelling and design*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

19. Sinderen, M., Joosten, S. and Farias, C. R. G.: Workflow Automation Based on OSI Job Transfer and Manipulation. To appear in *Computer Standards & Interfaces*, 1999.

20. Teege, G.: Object-Oriented Activity Support: A Model for Integrated CSCW Systems. *Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing*, 5(1), pp. 93-124, 1996.

Appendix A: Travel Claim System

These appendixes illustrate the application of our conceptual framework in the modelling of some cooperative applications. For each example we first describe informally the problem and represent this problem using our model. The representation of these examples makes uses of one or more object diagrams.

To prevent the duplication of some components, such as actors, services and information, we often place the association between these elements and their associated activity into top-level activities. In this way we do not burden the figures with too many details and preserve the correctness of the modelling.

Problem characterisation

This example aims at demonstrating the capability of our model for representing workflow systems. It consists of representing the Travel Claim process reported on [9].

The travel claim process consists of two process steps, viz. *Submit travel claim* and *Approve travel claim*, and the process element *Reimburse client*. The process element *Reimburse client* is further decomposed into process elements *Return approved travel claim to client* and *Transfer money*. Process steps are implemented by applications, in this case the Travel Claim System, while process elements are implemented by process steps or by process elements.

The execution of process steps *Submit travel claim* and *Approve travel claim* and the process element *Reimburse client* is responsibility of the agents client, manager and financial clerk, respectively. In order to execute these processes some information is required, viz. trip identification, client identification and travel claim identification.

Problem representation

The representation of the travel claim system using our model is quite simple. Figure 7 depicts the object diagram of the travel claim system.

The process steps *Submit travel claim* and *Approve travel claim* were represented using actions while the process element *Reimburse client* was represented using an activity. The decomposition of the process element *Reimburse client* into the process elements *Return approved travel claim to client* and *Transfer money* was modelled using actions. The relationship among these elements is simple and consists of associations whose type is *enable*, e.g., the completion/execution of the action "*Submit travel claim*" enables the execution of the action "*Approve travel claim*".

The Travel Claim System is the computerised service that supports the activities and actions identified erstwhile. Different actors perform these actions and activities. Client, manager and financial clerk are the roles assumed by these actors in this example. The information used by these activities was directly mapped using the information concept.

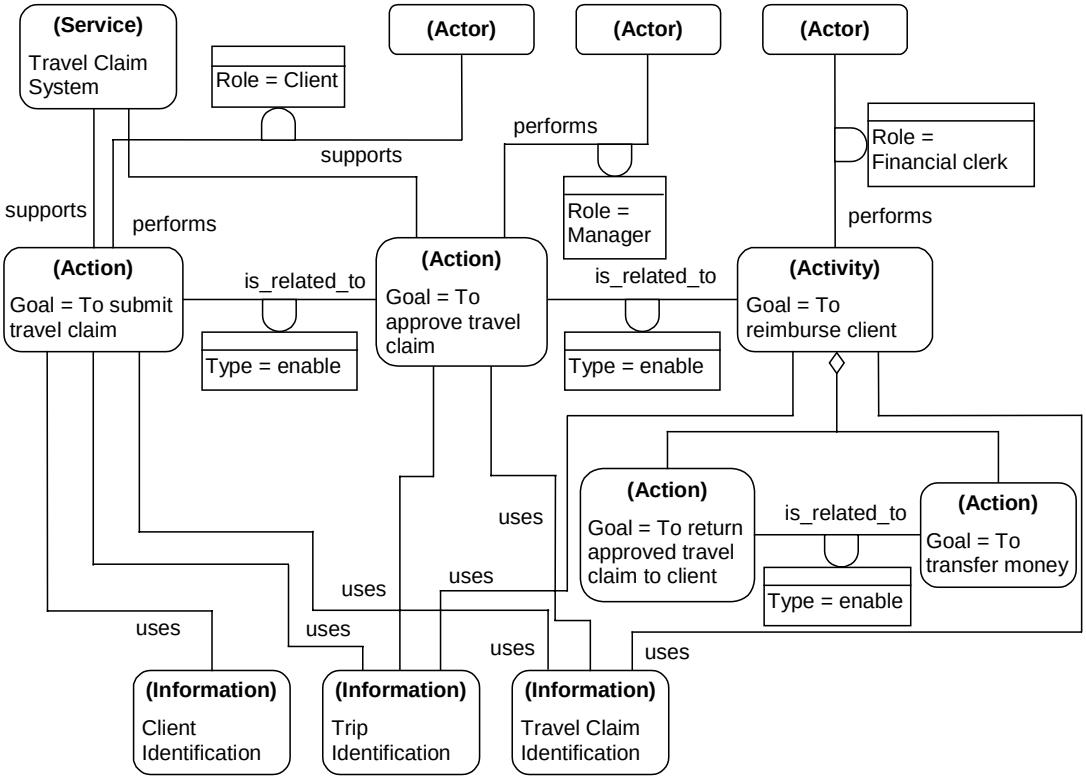


Figure 7. Travel Claim process object diagram.

Appendix B: Paper reviewing application

Problem characterisation

A generic paper reviewing application is as cooperative tool where the program conference committee and the paper reviewers work together to select the papers to be accepted in a conference or workshop.

One of the authors of a paper, called contact author, submit electronically a paper to the conference. When the submission deadline is reached all the papers received so far are distributed among the program committee members by the program committee chair. Several papers can be assigned to one committee member, however, for the sake of simplicity, each paper is assigned to only one committee member, which is responsible for the final evaluation of this paper. Papers received after the submission deadline are discarded and a rejection message is automatically sent to the contact author.

For each paper, the committee member responsible for it designates two or more reviewers, provided that each paper must have at least three reviews. Committee members may also review papers. In this case they assume the role of reviewers.

The review follows the criteria established in the review form. A reviewer must fill the review form in order to generate a review. After the reviewer has completed a review, she may check the other reviews of that paper. A reviewer can change her review at any time. This policy aims at stimulating a consensus between the reviewers. However, if a consensus is not reached, i.e., conflicting rates appear in the end of the reviewing process, the program committee member responsible for the paper together with the paper reviewers use some kind of conference tool to communicate with each other to try to reach an agreement. If no agreement is reached, the committee member responsible for that paper decides on a final rate for the paper herself.

When the reviewing phase finishes the program committee has to select the papers that are accepted. Using a conference tool, they choose, e.g., the minimum rate to a paper be approved, or the number of papers that will be approved. The contact author of each paper receives via e-mail the evaluation of her paper, together with further instructions if the paper has been accepted for the conference.

Problem representation

The paper reviewing process can be roughly split into three major activities, viz. "*To receive submitted papers*", "*To review these papers*" and "*To send paper evaluations*".

Figure 8 shows the high level structure of the paper reviewing application. This representation contains the three activities mentioned above, their relationship and the support of the paper reviewing application. The grey objects mean that these objects will be unfolded elsewhere. The completion or the partial execution of one activity enables the beginning of another. This is represented in Figure 8 as a relation between activity objects.

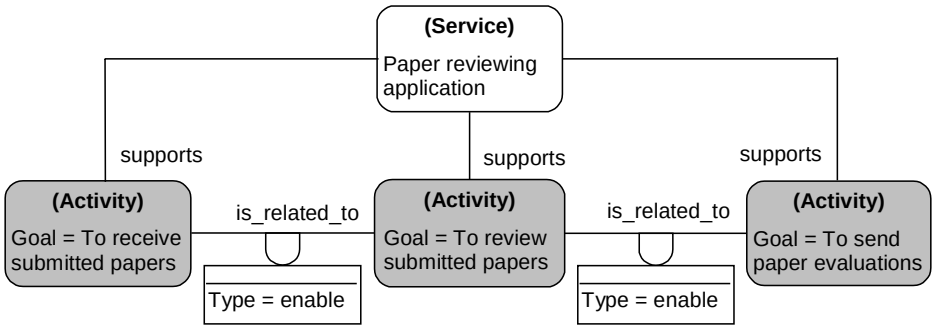


Figure 8. Paper reviewing application high level object diagram.

Figure 9 shows the modelling of the activity "*To receive submitted papers*". This activity aims at receiving and storing the papers submitted to the conference. The activity consists of two sub-activities, viz. "*To store upcoming papers*", which receives and store the submitted papers together with additional information about the authors, and "*To reject new papers* ", which rejects upcoming papers after the submission deadline has been reached. The completion of former sub-activity (submission deadline expired) enables the execution of latter. An autonomous actor performs the activity "*To receive submitted papers*", which is supported by both an e-mail service and the paper reviewing system.

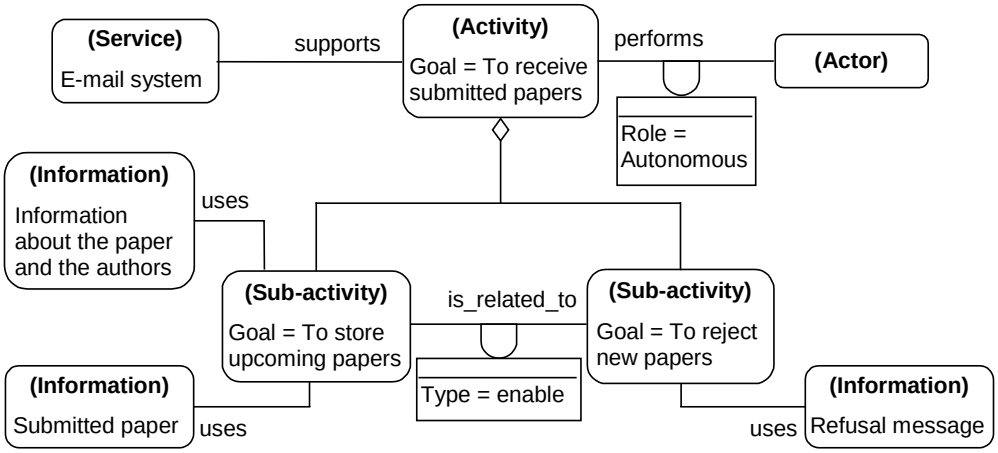


Figure 9. "To receive submitted papers" activity object diagram.

Figure 10 depicts the modelling of the activity "*To review submitted papers*". This activity consists of a number of sub-activities and actions, viz. "*To distribute papers among committee members*", "*To assign reviewers to a paper*", "*To review a paper*" and "*To decide on accepted papers*". The shadow objects represent several instances of the same elements. The choice of using actions or sub-activities in the model is primarily based on time duration and complexity. This activity is supported by a generic conferencing system and the paper reviewing system.

The action "*To distribute papers among committee members*" is performed by the committee chair and results in a list of assignments (paper x committee member). The completion of this action enables the beginning of the action "*To assign reviewers to a paper*". One instance of the action "*To assign reviewers to a paper*" is created for each paper submitted to the conference. This action is performed by the committee member responsible for the paper and results in another list of assignments (paper x reviewer).

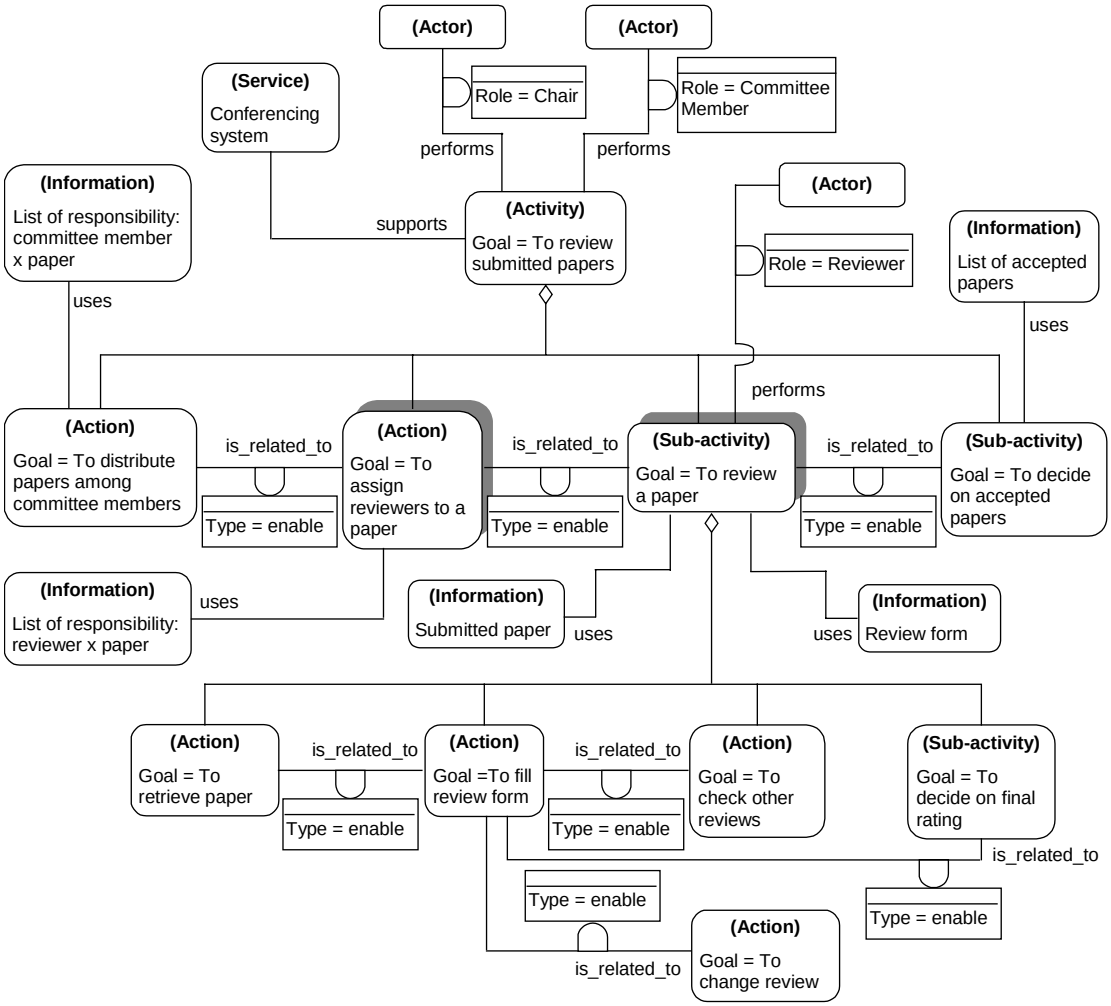


Figure 10.“To review submitted papers” activity object diagram.

The completion of the action “*To assign reviewers to a paper*” associated with a specific paper enables the next phase of the reviewing process, i.e., the reviewing of the paper itself, which is represented by the sub-activity “*To review a paper*”. This sub-activity is performed by reviewers and by the committee member responsible for that paper, and makes uses of two kind of information, viz. the submitted paper and the review form. The activity “To *review a paper*” consists of four actions and one additional sub-activity. We refrain from discussing this activity any further for the sake of conciseness.

The completion of all sub-activities “*To review a paper*” enables the execution of the next and last sub-activity of the activity “*To review submitted papers*”, viz. the sub-activity “*To decide on accepted papers*”. This sub-activity is performed by the program committee members and the committee chair and results in a list of accepted papers. This sub-activity is not further decomposed in this example because a similar example is developed in Appendix C.

Figure 11 depicts the representation of the activity “*To send paper evaluation*”.

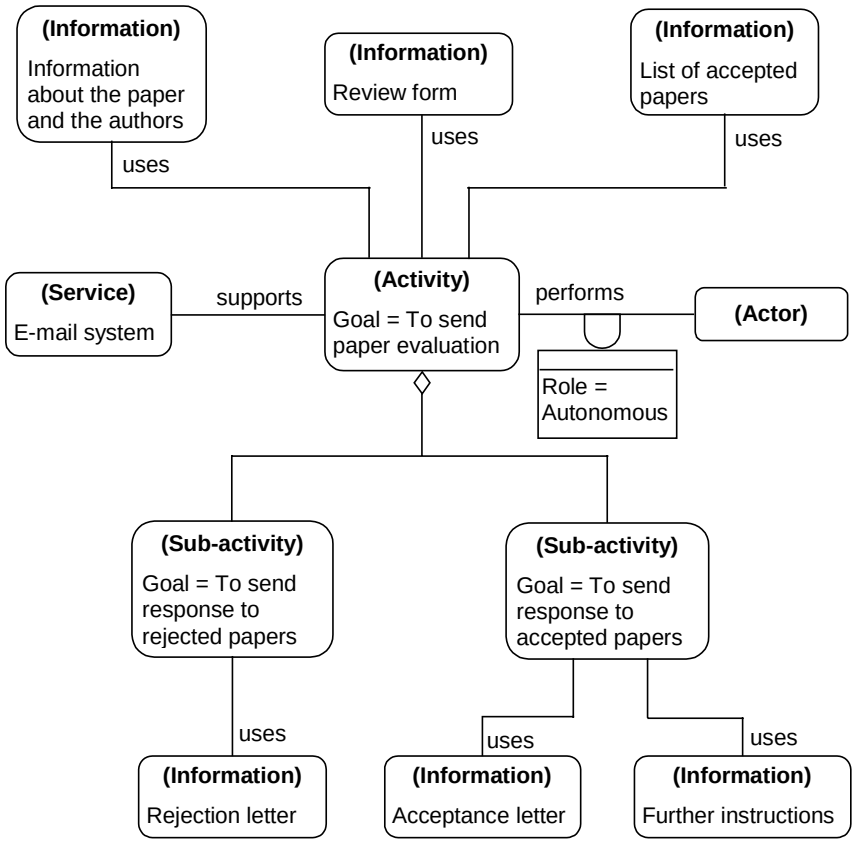


Figure 11. "To send paper evaluation" activity object diagram.

An autonomous actor performs the activity "*To send paper evaluation*", which is supported by an e-mail system and the paper reviewing system. This activity consists of two independent sub-activities, viz. "*To send response to rejected papers*" and "*To send response to accepted papers*". The former sub-activity aims at sending the rejection notice and the review forms to the contact author of a rejected paper, while the latter aims at sending the acceptance notice, the review forms and further instructions to the contact author of an accepted paper.

Appendix C: Conferencing system

Problem characterisation

In this example we consider a general brainstorming conferencing system. A brainstorming conference may be roughly split into four phases, viz. *Choosing the participants*, *Joining the conference*, *Brainstorming* and *Voting*. During the phase of choosing the participants, a mediator invites all the possible participants of the conference. The mediator may use any kind of method to invite the conference participants, such as sending an e-mail, telephoning, putting a note in a notes board and even inviting the participant by herself.

A previously invited participant can join the conference at any time. Thus, support for latecomers must be provided. After joining the conference the new conference member receive the status of the conference and is able to participate in the brainstorming and voting phases.

In the brainstorming phase all the conference members and the mediator can make issues and respond to issues. When responding to an issue, it is possible to either support the issue or object to the issue. Floor control policies should be used to coordinate the order of the statements. At any time during the brainstorming phase the mediator can decide to vote for certain proposals switching the conference to the voting phase. A voting proposal is elaborated and all the conference members can vote for the proposal, against the proposal or even refrain from voting. After the voting the counting is performed and the mediator decides on elaborating a new voting proposal, returning to the brainstorming phase or ending the conference.

Problem representation

The brainstorming conferencing system can be split into the following major activities, viz., "*To choose the participants*", "*To control the access to the conference*", and "*To participate in the conference*", which is split into the sub-activities "*Brainstorming*" and "*Voting*".

Figure 12 shows the high level structure of the brainstorming conferencing system. This representation contains the activities mentioned previously, their relationship and the support provided by the brainstorming conferencing system. The completion of the activity "*To choose the participants*" enables the activity "*To control the access to the conference*". When the first participant joins the conference, the activity "*To participate in the conference*" is enabled. On the other hand, when the last participant leaves the conference, the acitivity "*To participate in the conference*" is disabled. This cenario provides the means for implementing a synchronous or asynchronous conference.

Figure 13 depicts the object diagram of the activity "*To choose the participants*". This activity aims at choosing and inviting the participants of the conference and consists of two related sub-activities, viz., "*To elaborate the list of participants*" and "*To invite the participants*". The former sub-activity is performed by the mediator, while the latter is performed by an autonomous agent. The activity "*To choose the participants*" is also suportted by an e-mail system and produces a list of possible participants.

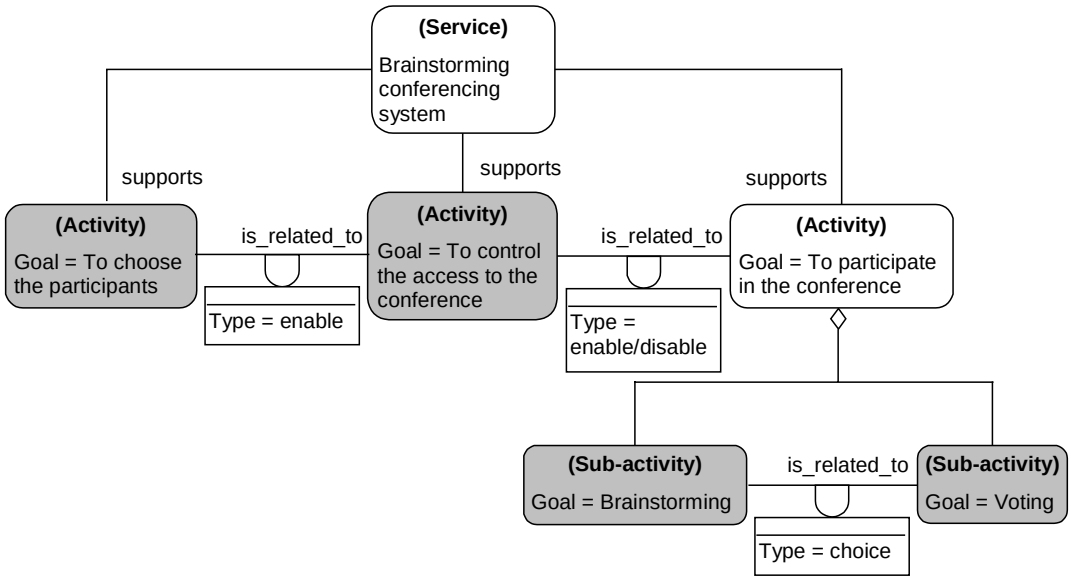


Figure 12. Conference system object diagram.

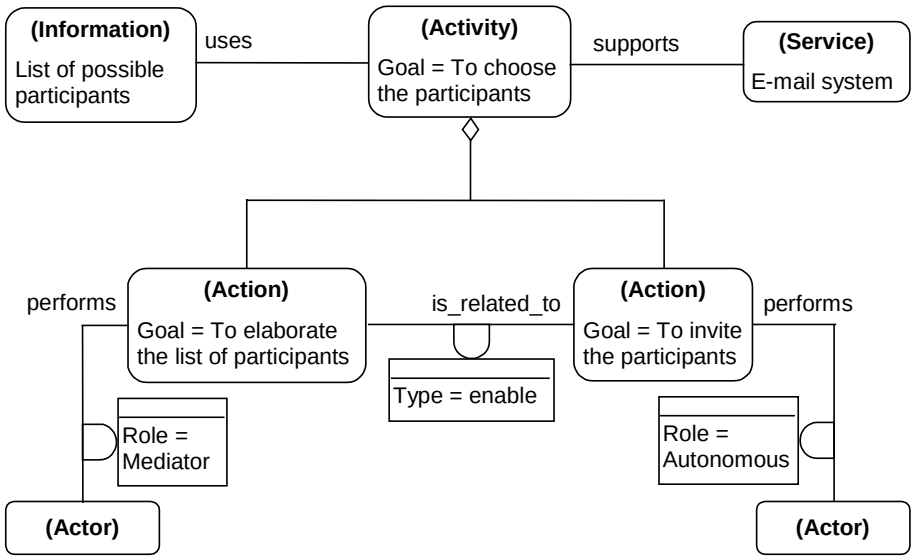


Figure 13. "To choose the participants" activity object diagram.

Figure 14 shows the object diagram of the activity "*To control the access to the conference*". This activity aims at provides the access to a participant when it joins the conference and release this access when the participant leaves the conference. This activity consists of the following sub-activities, viz., "*To control conference joining*", "*To control conference leaving*" and "*To control the list of conference members*". Both the participants of the conference and the mediator of the conference join the conference by "executing" the sub-activity "*To control conference joining*", which is unfolded in Figure 15, and leave the conference performing the sub-activity "*To control conference leaving*". Actually the mediator/participant perfoms the action "*To leave the conference*". Every time a participant join or leave the conference, the sub-activity "*To control the list of conference members*" is enabled, which either adds or removes the mediator/participant from the list of current conference members. An autonomous actor perfoms several sub-activities of the activity "*To control the accesss to the conference*". However, only the relatioship between the actor and the top activity is represented.

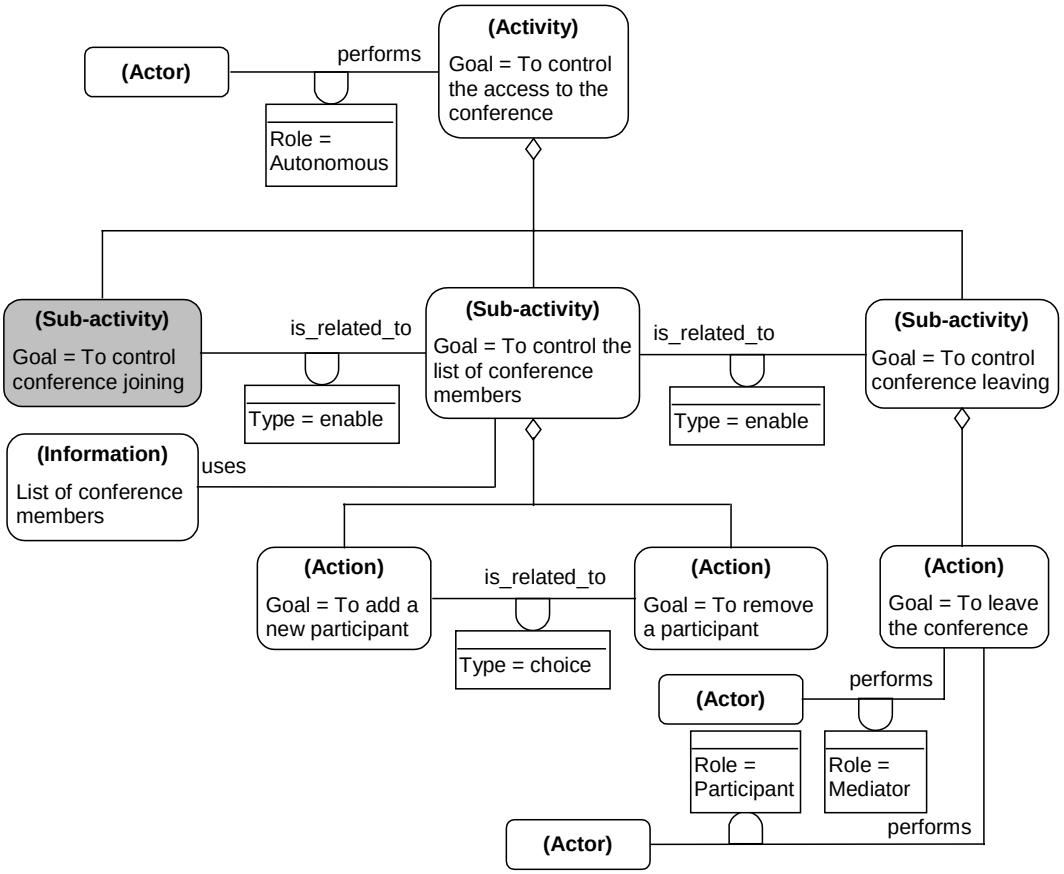


Figure 14. "To control the list of conference members" activity object diagram.

Figure 15 depitcs the object diagram of the sub-activity "*To control conference joining*". This activity is decomposed into three actions, viz., "*To join the conference*", "*To check is the incoming participant belongs to the conference*" and "*To send conference status to new participant*". The action "*To join the conference*" is performed by the mediator and the participants in order to join the conference. The execution of this action anables the action "*To check is the incoming participant belongs to the conference*", which is performed by an autononous actor. This action makes use of the list of the list of possible participants to check whether or not an incoming participant belongs to the conference. If this participant is alowed in the conference the action "*To send conference status to new participant*" is enabled. The action "*To send conference status to new participant*" is also performed by an autonomous agent and simply sends the conference status to the new participant.

Figure 16 presents the object diagram of the sub-activity "*Brainstorming*". This activity is decomposed into the actions "*To make an issue*" and "*To update conference status*" and into the sub-activity "*To respond to an issue*". The sub-activity "*To respond to an issue*" is further decomposed into the actions "*To support the issue*" and "*To object to the issue*". Both the meadiator and the participants can make issues and respond to issues. After the execution of any of these activities, an autonomous actor performs the action "*To update conference status*" to update the conference status. When responding to an issue, the mediator or the participants can either support or object to the issue. A choice is made between the actions "*To support the issue*" and "*To object to the issue*".

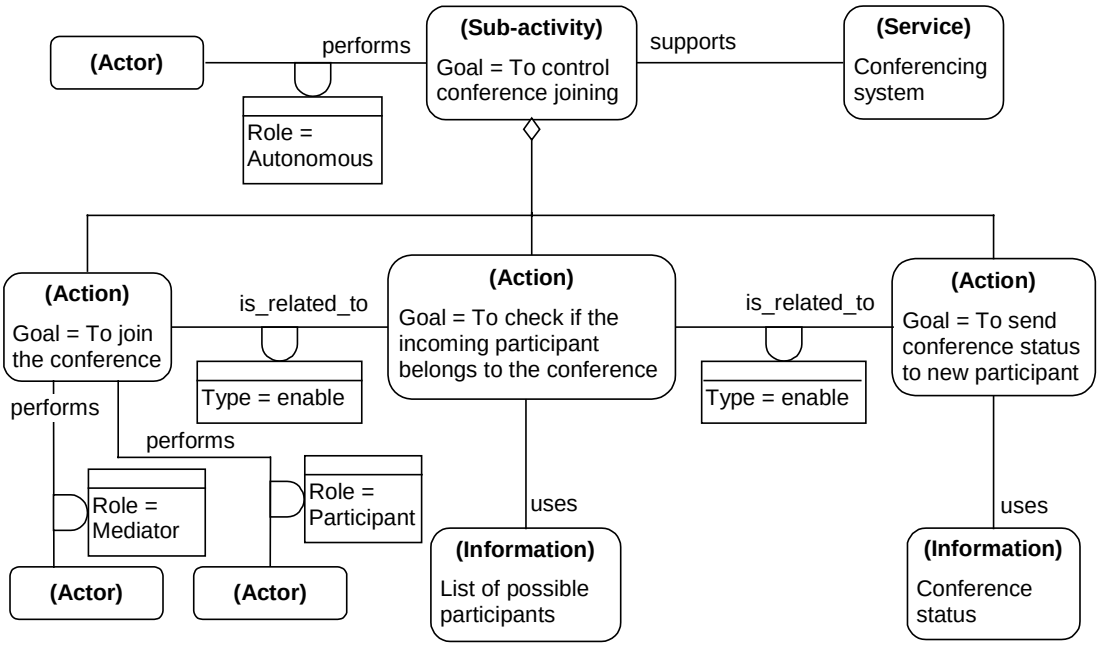


Figure 15. "To control conference joining" activity object diagram.

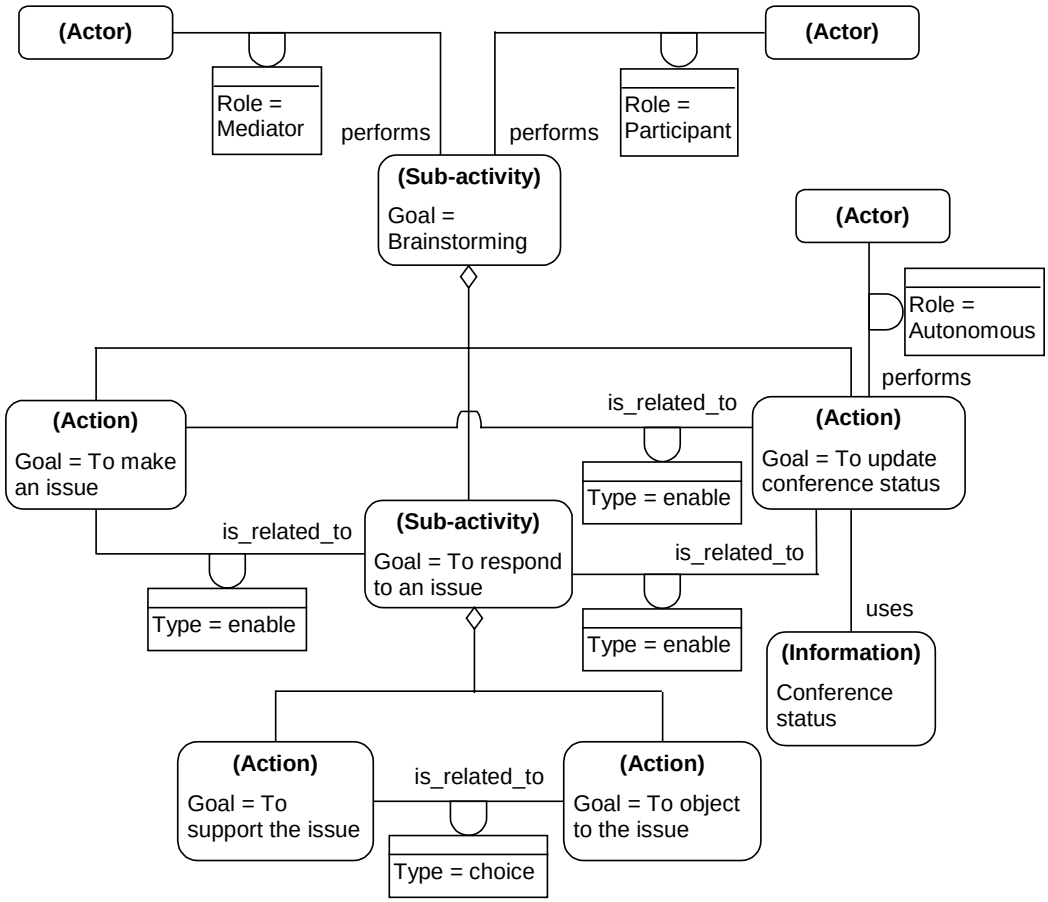


Figure 16. "Brainstorming" activity object diagram.

Figure 17 shows the object diagram of the sub-activity "*Voting*". This activity consists of the actions "*To elaborate voting proposal*" and "*To count the votes*" and the sub-activity "*To vote the proposal*". The sub-activity "*To vote the proposal*" is further decomposed into the actions "*To vote for the proposal*", "*To vote against the proposal*" and "*To refraim from voting*". The mediator performs the action "*To elaborate voting proposal*" in order to elaborate a voting proposal and request a voting activity. The execution of this action enables the sub-activity "*To vote the proposal*". Both the mediator and the participants perform this activity and execute one of the following actions: "*To vote for the proposal*", to vote for the proposal; "*To vote against the proposal*", to vote against the proposal; and "*To refraim from voting*", to refraim from voting. After the activity "*To vote the proposal*" has finished an autonomous actor performs the action "*To count the votes*" to count the votes and update the conference status.

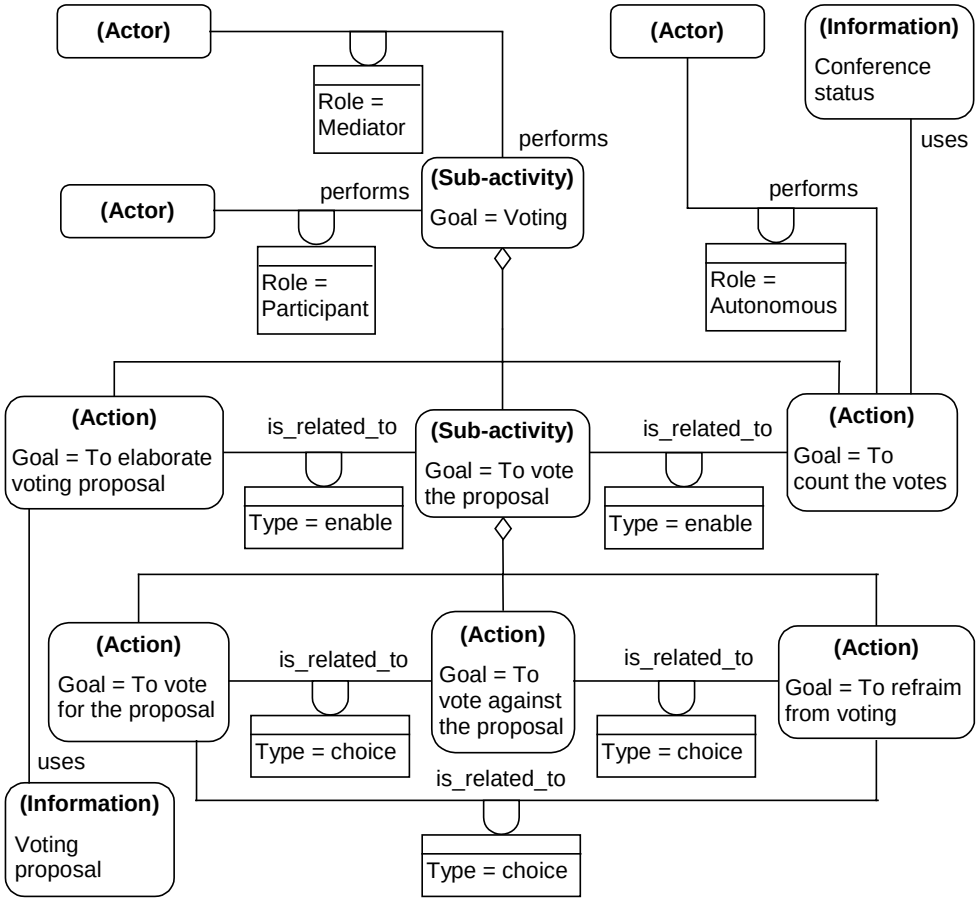


Figure 17. "Voting" activity object diagram.