Dynamic Institutions for Teamwork

Mario Gómez and Enric Plaza
mgomez@csd.abdn.ac.uk, enric@iiia.csic.es

[1] Department of Computing Science, University of Aberdeen
[2] Artificial Intelligence Research Institute, Spanish National Research Council

Abstract. We present a dynamic electronic institutions approach for teamwork. In this model, agent teams are designed and deployed on-the-fly so as to met the requirements of the task at hand. The result is a new form of electronic institution that is created dynamically out of existing components. We also introduce a case-based learning mechanism to form new agent teams by reusing complete or partial team-designs used in the past for similar problems.

1 Introduction

Cooperative problem solving (CPS) is a form of social interaction in which a group of agents work together to achieve a common goal. Several models have been proposed to account for this form of interaction from different perspectives: distributed artificial intelligence, economics, philosophy, organization science and social sciences. From the artificial intelligence perspective there are two main approaches to cooperation: a micro-level –agent-centered– view, which is focused on the internal architecture or the decision-making model of individual agents, and a macro-level –social– view, which is focused on the societal and organizational aspects of cooperation. Most of the models and theories of cooperation proposed for MAS have adopted the agent-centered view, typically based on some refinement of the beliefs, desires and intentions (BDI) paradigm, such as the *Joint Intentions* model [12] and the *SharedPlans* theory [9].

Some of the most challenging issues faced by the MAS community are related to the creation of open MAS [11]. Closed systems are typically designed by one team for one homogeneous environment, while in open MAS the participants (both human and software agents) are unknown beforehand, may change over time and may be developed by different parties. Therefore, those infrastructures that adopt a social view on cooperation seem more appropriate that those adopting a micro-level view, for the former do not enforce a particular agent architecture.

In addition, some aspects of complex system development become more difficult by adopting an agent-centered approach: since agents are autonomous, the patterns and the effects of their interactions are uncertain, and it is extremely difficult to predict the behavior of the overall system based on its constituent components, because of the strong possibility of emergent behavior [10]. These problems can be circumvented by restraining interactions and imposing preset

organizational structures, which are characteristic of the social view. The Civil Agent Societies framework [2] and the electronic institutions formalism [13, 15, 5] are good examples of this approach; many others can be found in the COIN international workshop series on coordination, organizations, institutions, and norms [1].

An electronic institution (or e-Institution) refers to a sort of "virtual place" that is designed to support and facilitate certain goals to the human and software agents concurring to that place by establishing explicit conventions. Since these goals are achieved by means of the interaction of agents, an e-institution provides the social mediation layer required by agents to achieve a successful interaction: interaction protocols, shared ontologies, communication languages and social behavior rules.

All in all, a main issue arises when trying to use preset organizational structures to operationalize CPS: the need for different team structures to deal with different problem types. The e-institutions formalism was originally conceived to formalize and implement static organizations of agents; therefore, at first glance it seems inadequate to use such a formalism for flexible teamwork. In this paper we introduce a proposal that uses the e-institution formalism in a novel way: dynamic institutions for teamwork. These institutions are created on-the-fly out of existing components that capture the communication and coordination aspects of teamwork.

The paper is structured as follows: Section §2 puts our institutional model of teamwork in context by introducing the framework this model is part of, §3 describes our proposal to model teamwork using the e-Institutions formalism [4], §4 describes a technique to improve team design by using case-based reasoning, and finally, §5 summarizes our contributions.

2 The ORCAS framework

In this paper we present an institutional approach to CPS that is part of the ORCAS framework for developing and deploying cooperative MAS [6]. The main contributions of this framework are:

- An *agent capability description language* (ACDL) that supports all the activities required to cooperate in open environments, from the discovery and invocation of capabilities, to their composition and coordination.
- A model of CPS that is driven by the specification of requirements for every instance of a problem to be solved
- An agent platform for developing and deploying cooperative MAS in open environments

Figure 1 depicts the main components of the ORCAS ACDL, and the activities enabled by each component. An agent provides one or more capabilities. There are two types of capability: *skill* and *task-decomposer*. Skills are primitive, non decomposable capabilities, while task-decomposers decompose a problem (a task) into more elementary problems (subtasks), so as to solve complex problems

that primitive capabilities cannot accomplish alone. The *knowledge-level description* of a capability specifies features such as the input, output, preconditions, and postconditions, which can be used by middle agents to discover and compose capabilities. However, in order to interact with the provider of a given capability (to invoke the capability, pass input data and get the results back), the requester agent must use an interaction protocol that is compatible with the capability of interest and is supported by its provider. In ORCAS this interaction protocol is referred to as the *communication* of a capability (take note that the same capability could be invoked using different protocols). Finally, the information required to coordinate multiple agents that are cooperating to solve a problem together is specified by the *operational description* of a task decomposer, which describes the control flow among subtasks (sequencing, parallelism, choices, etc.) in terms of agent roles.

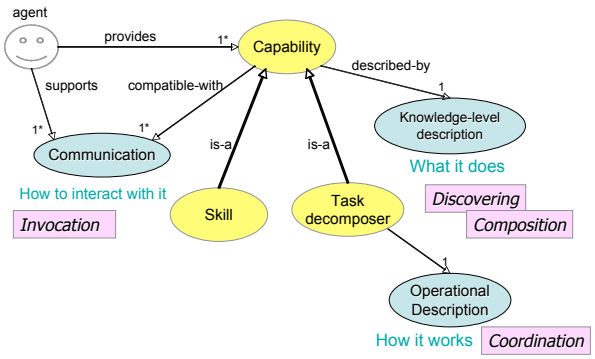


Fig. 1. Overview of the ORCAS ACDL

The ORCAS platform provides all the infrastructure required by agents to successfully cooperate according to the ORCAS model for CPS, which is depicted in Figure 2. The *problem specification* process produces a specification of problem requirements to be met by a team, including a description of the application domain (a collection of domain models) and the problem data to be used during teamwork. The *team design* process uses the problem requirements to build a *task-configuration*, which is a knowledge-level specification of: (1) the tasks to solve, (2) the capabilities to apply, and (3) the domain knowledge required by a team of agents in order to solve a given problem according to its specific requirements. The resulting task-configuration is used during *team formation* to allocate tasks and subtasks to agents, and to instruct agents on how solve the problem cooperatively. Finally, during *teamwork*, team members try to solve the problem together by following the instructions received during team formation, thus complying with the specific requirements of the problem at hand. To note that the ORCAS model for CPS should not be understood as a fixed sequence of

steps, instead, we have implemented strategies that interleave team design and team formation with teamwork. These strategies enable the reconfiguration of agent teams dynamically so as to react to agent failure and other changes in the environment.

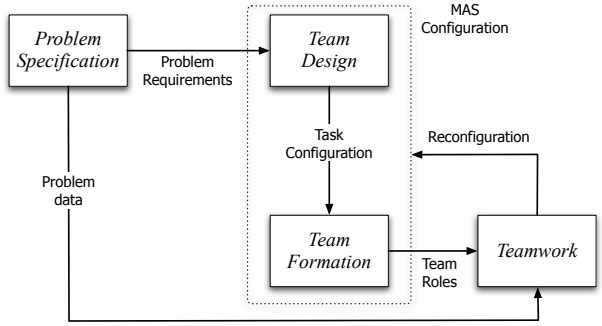


Fig. 2. The ORCAS model for the cooperative problem solving process.

It should be remarked that, within the ORCAS framework, the e-institutions formalism is used in two ways: on the one hand, we use concepts adapted from the ISLANDER e-institutions formalism [4, 3] for specifying some elements of the ORCAS ACDL (the communication and the operational description); on the other hand, the ORCAS agent platform is itself an e-institution that provides mediation services for both providers and requesters of problem solving capabilities to successfully cooperate.

The knowledge-level description of a capability and the mechanisms used in ORCAS to discover and compose capabilities (which are part of the team design process) have been described elsewhere [7]. The ORCAS agent platform is described in [8]. In this paper we focus on those aspects of the ORCAS ACDL that are based on the e-institutions formalism, namely the communication and the operational description, and how are these elements used to represent the interaction and coordination requirements of teamwork. These are the subjects of the following section.

3 Dynamic institutions for hierarchical teamwork

The ORCAS ACDL specifies the communication and operational description of capabilities using elements from the ISLANDER formalism in a novel way, so it seems appropriate to briefly review the main concepts of this formalism before describing their use in ORCAS:

1. *Agent roles:* agents are the players in an e-institution, interacting by the exchange of speech acts, whereas roles are standardized patterns of behavior required by agents playing part in given functional relationships.

2. *Dialogic framework:* determines the valid illocutions that can be exchanged among agents, including the vocabulary (ontology) and the agent communication language.
3. *Scenes:* a scene defines an interaction protocol among a set of agent roles, using the illocutions allowed by a given dialogic framework.
4. *Performative structure:* a network of connected scenes that captures the relationships among scenes; a performative structure constrains the paths agents can traverse to move from one scene to another, depending on the roles they are playing.

In ORCAS the specification of capabilities at the knowledge level enables the automated discovery and composition of capabilities, without taking into account neither the communication aspects required to invoke a capability, nor the operational aspects required to coordinate the behavior of several agents. These features are specified in the ORCAS ACDL adapting concepts from ISLANDER, as follows:

Communication: specifies one or several interaction protocols that can be used to interact with agent to invoke a given capability and get back the result of applying it. This feature is specified using the notion of *scene* taken from the e-institutions formalism.

Operational Description: specifies the control flow among the subtasks introduced by a task-decomposer, using a modified version of the *performative structure* concept from the e-institutions formalism.

A team in ORCAS is designed to solve a problem represented by a knowledge-level structure referred to as a *task-configuration* (the reader is referred to [7] for a more detailed description). Figure 3 shows an example of a task-configuration for a task called *Information-Search*. This task is decomposed into four tasks by the *Meta-search* task-decomposer: *Elaborate-query*, *Customize-query*, *Retrieve* and *Aggregate*, which is further decomposed by the *Aggregation* capability into two subtasks: *Elaborate-items* and *Aggregate-items*. The example includes some skills requiring domain knowledge: the *Query-expansion-with-thesaurus* requires a thesaurus (e.g. *MeSH*, a medical thesaurus), and the *Retrieval* and *Query-customization* skills require a description of information sources.

Any ORCAS team follows the hierarchical structure of a task-configuration, with one team-role per task. In particular, each team role includes the following elements: a team-role identifier (the same task could appear multiple times in the same task-configuration, so a unique team-role identifier is required), the identifier of a task to be solved, the identifier of a capability to apply, the domain knowledge to be used by the selected capability (if needed), and optionally, if the capability is a task decomposer, the information required to delegate subtasks to other team-members, which includes, for each subtask: the team member selected to play the task (or several agents in the case of tasks to be performed multiple times in parallel), a collection of reserve agents to use in case that the selected team member fails, and a communication protocol that is compatible with the selected capability and shared by both the agent assigned to the parent

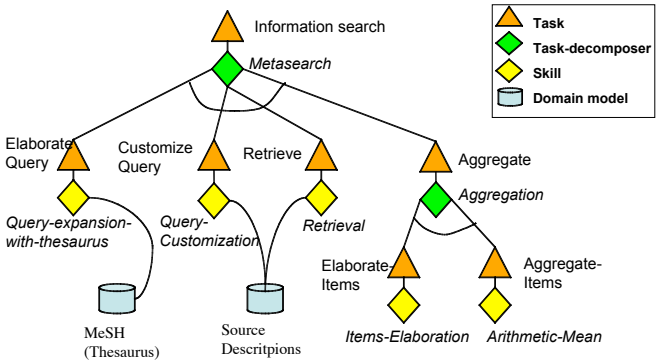
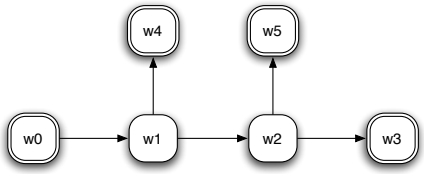


Fig. 3. Task-configuration example

task, and the agent or agents assigned to the subtask. Next subsections address, respectively, the specification of the communication and operational description of a capability in ORCAS.

3.1 Communication

Agent capabilities should be specified independently of other agents in order to maximize their reuse and facilitate their specification by third party agent developers. In the general case, agent developers do not know a priori the tasks that could be achieved by a particular capability, neither the domains they could be applied to. As a consequence, the team roles an agent could play using a capability are not known in advance, thus the scenes used to specify the communication requirements of an agent over certain capability cannot be specified in terms of specific team-roles, but in terms of abstract, generic problem solving roles. Since ORCAS teams are designed in terms of a hierarchical decomposition of tasks into subtasks, teamwork is organized as a hierarchy of team-roles.



1. request (?x Coordinator) (?y Operator) (perform ?team-role ?input)
2. agree !y !x (!team-role !Input)
3. inform !y !x (?team-role ?output)
4. refuse !y !x (!team-role !Input)
5. error !y !x! (team-role !Input)

Fig. 4. Example of a communication scene

Some positions within a team (team-roles) are bound to a task-decomposer, thus the agents playing those team-roles are responsible of delegating subtasks to other agents, receiving the results, and performing intermediate data processing between subtasks. In such an scenario, we establish an abstract communication model with two basic roles: *coordinator*, which is adopted by an agent willing to decompose a task into subtasks, and *operator*, which is adopted by the agent having to perform a task on demand, using the data provided by another agent that acts as coordinator of a top-level task

Figure 4 shows a scene depicting the communication requirements of an agent over a capability by using a typical request-inform protocol in terms of our two generic roles: *Coordinator* and *Operator*. Symbol *?* denotes a new bind for a variable, while *!* denotes a variable that has been already bound to a value.

3.2 Operational description

The operational description of a task decomposer is used to specify the coordination among agents in terms of the role-flow policy and the control flow among subtasks. Figure 5 depicts some of the control flow constructions allowed by a performative structure: (a) tasks performed consecutively, in sequence; (b) choice between alternative courses of action; (c) tasks performed in parallel; and (d) tasks that can be executed multiple times.

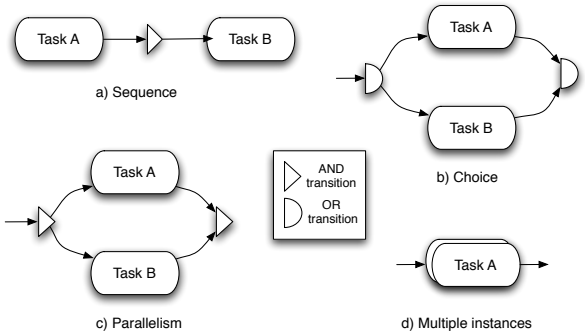


Fig. 5. Control flow among subtasks used in operational descriptions

In ORCAS the operational description of a task-decomposer is based on performative structures, with some distinctive features: as in the e-institutions formalism, each ORCAS scene within a performative structure must be instantiated by a communication protocol (except the *Start* and *End* scenes). However, in ORCAS the scenes within a performative structure are not instantiated beforehand; that is to say, they are not bound to a specific communication protocol. Instead, the scenes of an operational description are instantiated during team

formation, using as a source the set of communication protocols shared by the agents having to interact.

After instantiation, each scene in an operational description corresponds to the communication required to solve a subtask, which implies an agent acting as coordinator invoking the capability provided by another agent acting as operator (or several operators in the case of multiple-instantiated tasks). The coordinator and the operators must use the same communication protocol in order to successfully communicate. Consequently, the instantiation of the scenes in an operational description is done using only those communication protocols shared by the agents involved in a scene. To note that team members are selected during team formation, and thus the set of shared communication protocols is not known until the team members are decided.

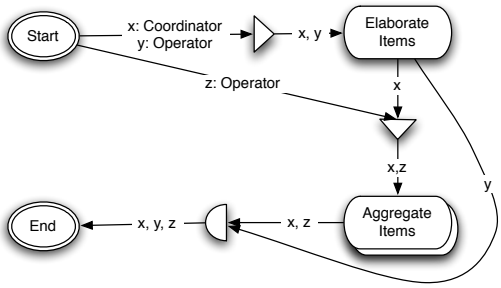


Fig. 6. Example of an operational description

Figure 6 shows an example of an operational description for a task-decomposer called Aggregation. This task-decomposer introduces two subtasks: Elaborate-items (EI) and Aggregate-items (AI). Thus, the operational description has two main scenes, one for each subtask, and three role variables: x is a coordinator role, to be played by the agent applying the task-decomposer; y and z are both operator roles; y participates in EI, and z participates AI. Notice that the coordinator (x) is the same in both scenes; it enters EI first and moves to AI only after EI ends.

Since each task-decomposer has an operational description, and the ORCAS organization of a team follows the hierarchical decomposition of tasks into subtasks that results of applying task-decomposers, we can model the operational description of a complete team as nested structure of operational descriptions.

Figure 7 depicts the operational description of a team. The top team-role, associated to the Information Search task, is bound to a task-decomposer (Meta-Serach) that introduces three subtasks: Customize Query, Retrieve and Aggregate. Therefore, the top team-role will follow an operational description that contains three scenes, one for each subtask. In addition, the last of these subtasks is bound to another task-decomposer, Aggregation, which in turn introduces a new

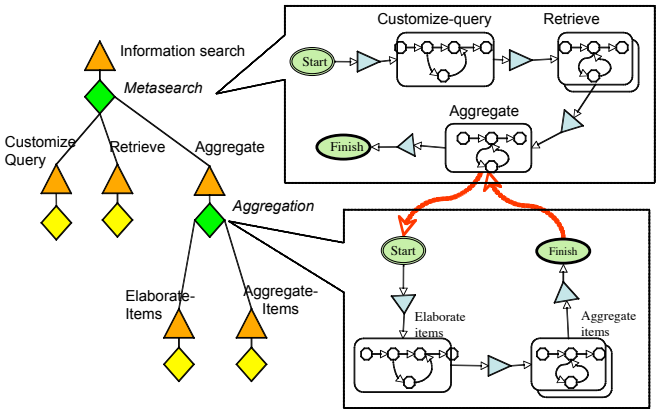


Fig. 7. Teamwork as a nested structure of operational descriptions

operational description. The new operational description is nested to the team leader's operational description, and has scenes, one for every subtask: Elaborate-Items and Aggregate-Items.

Teamwork follows the control flow and the communication scenes established by the nested structure of operational descriptions associated to task-decomposers (already instantiated during team formation). Each scene within an operational description refers to a communication protocol to be played by two agents, one applying a task-decomposer and playing the coordinator role, and one assigned to the corresponding subtask playing the operator role. When an agent playing an operator role has to apply itself a task-decomposer, it will follow the associated operational description playing itself the coordinator role. The execution of an operational description does not finish until all the nested operational descriptions are executed.

Each time a new team is formed according to a task-configuration, a new structure of nested operational descriptions is composed and their scenes instantiated. We regard this structure as a dynamic institution, since it is configured on-the-fly, out of the communication protocols and the operational descriptions supported by the selected team members.

4 Learning Team Designs

The ORCAS description of *team designs* allows that reasoning and learning processes may be applied to them by an agent capable of convening a team. For this purpose, an agent-oriented case-based reasoning (CBR) technique called CoopCA has been developed based on the notion of *compositional cases* [14]. In team design, a case is a pair (P, S), where the problem P is the specification of the *task* a team should be able to achieve and the solution S is the *task-configuration* of the team; a case is a compositional case when the solution is a configuration of

components —as the ORCAS ACDL components in team design. Notice that a task basically specifies two conditions: the *assumptions* (those properties that are assumed to be true in the world) and *goals* (those properties that are to be achieved by the team).

Learning in CBR allows agents to solve much faster routine and easy tasks. An agent using CoopCA will store in its case base those team designs the agent has convened in the past. Since in a real environment regularities are common, an agent will encounter both routine tasks and novel tasks. When a new task that needs a team to be convened is equal or similar to tasks the agent has solved in the past using specific team designs, case-based reasoning will reuse the solutions of the past to fit the current situation: a previous team design will be used, possibly with a few alterations to adapt the new team to the differences (in assumptions and/or goals) between the old task and the current task.

When a new task is novel, in the sense that it is rather different from any other task previously solved by the agent, CoopCA is capable of achieving a new team design from scratch. However, CoopCA derives a new task-configuration in a search process that is guided by the past cases, and is able to find, for instance, that a specific *subtask* was solved in the past by a particular team, and will incorporate it as a *subteam* for that subtask in the overall team design. Thus, the agent playing the role of convener can learn from experience about the particular *team institutions* that achieve certain tasks. Notice that this learning is developed at the institutional level: the agent learns that a specific team institution is able to achieve a certain task — does not learn about the performance of a specific team with concrete agents performing the institution's roles.

Finally, CBR is used for *dynamic reconfiguration* when some event precludes the usability of the current team institution. For instance, imagine that and agent that was supposed to perform team-role R_j goes offline or refuses to satisfy its previous commitment; and imagine there is no other available agent capable of satisfying the requirements of that role: under this conditions the task associated to R_j could not be achieved and the task-configuration that shaped the current team is no longer viable. The CBR process however can continue its search process to find another task-configuration (if it exists) that achieves the same overall task. There is no need to stick to a fixed design when several possible solutions are available.

5 Conclusions

In this paper, we have presented a novel approach to teamwork specification using concepts adapted from the e-Institutions formalism. In this approach the communication and coordination aspects required for teamwork are reusable components that are used by agents to specify their problem solving capabilities. By doing so, middle agents such as brokers and matchmakers can reason about the communication and coordination aspects of individual agents to dynamically build an e-Institution that supports flexible teamwork.

We adapt the electronic formalism to handle the dynamics of teamwork. While e-institutions are supposed to be static structures characterized by a predefined network of scenes (a performative structure), we conceive teamwork as a dynamic institution that is build on the fly out of existing components: operational descriptions and communication protocols. The operational description of a task-decomposer describes the control flow among subtasks using a specific kind of performative structure in which the communication scenes are not instantiated beforehand. The instantiation of those scenes is done at runtime by selecting communication protocols that are shared by the agents involved in a given scene. The result is a hierarchical model of teamwork represented by nested performative structures instantiated and composed on-the-fly during team formation.

By adapting the e-Institutions formalism for teamwork, we expect to bring in some of the benefits of the social-approach in general, and the e-institutions approach in particular: promoting the development of agents by third parties by avoiding the imposition of a specific agent architecture (favors openness); increasing the degree of control over the global system behavior; and making the system more predictable, which in turn fosters trustiness.

Finally, by introducing a case-based reasoning approach to team design we have enabled a learning process that speeds up the configuration of new teams by reusing previous team designs for solving new problems

Acknowledgements

This research was sponsored by the Spanish Council for Scientific Research under the MID-CBR (TIN 2006-15140-C03-01) project.

References

1. O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vaźquez-Salceda, editors. *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *Lecture Notes in Artificial Intelligence*. Springer, December 2006.
2. C. Dellarocas. Contractual Agent Societies. negotiated shared connote and social control in open multi-agent systems. In *Proceedings of the Workshop on Norms and Institutions in Multi-Agent Systems, ICMAS'02*, 2000.
3. M. Esteva. *Electronic Institutions: From Specification to Development*, volume 14 of *Monografies de l'Institut d'Investigació en Intel.ligència Artificial*. Spanish National Research Council, 2003.
4. M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In *Intelligent Agents VIII: Lecture Notes in Artificial Intelligence*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 348–366. Springer-Verlag, 2002.
5. M. Esteva, J. A. Rodriguez, C. Sierra, P. Garcia, and J. L. Arcos. On the formal specifications of electronic institutions. In *Agent-mediated Electronic commerce. The European AgentLink Perspective*, volume 1991 of *Lecture Notes in Artificial Intelligence*, pages 126–147, 2001.

6. M. Gómez. *Open, Reusable and Configurable Multi-Agent Systems: A Knowledge-Modelling Approach*, volume 23 of *Monografies de l'Institut d'Investigació en Intel.ligència Artificial*. Spanish National Research Council, 2004.
7. M. Gómez and E. Plaza. Extending matchmaking to maximize capability reuse. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, volume 1, 2004.
8. M. Gómez and E. Plaza. The ORCAS e-Institution: a Platform to Develop Open, Reusable and Configurable Multi-Agent Systems. *International Journal on Intelligent Control and Systems. Special Issue on Distributed Intelligent Systems*, juny 2007.
9. B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
10. N. R. Jennings. On-agent-based software engineering. *Artificial Intelligence*, 117:227–296, 2000.
11. M. Klein. The Challenge: Enabling Robust Open Multi-Agent Systems, 2000.
12. H. J. Levesque. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, 1990.
13. P. Noriega. *Agent-Mediated Auctions: The Fish-Market Metaphor*. PhD thesis, Universitat Autònoma de Barcelona, 1997.
14. E. Plaza. Cooperative reuse for compositional cases in multi-agent systems. *Lecture Notes in Computer Science*, 3620:382–396, 2005.
15. J. A. Rodríguez-Aguilar. *On the Design and Construction of Agent-mediated Electronic Institutions*. PhD thesis, Universitat Autnoma de Barcelona, 1997.