Integrating Knowledge Modeling and Multi-Agent Systems

Mario Gómez and Enric Plaza
Artificial Intelligence Research Institute - Spanish Scientific Research Council
Campus UAB
08193 Bellaterra
Barcelona, Spain
{mario, enric}@iiia.csic.es

Abstract

This paper outlines ORCAS, a framework for open Multi-Agent Systems (MAS) that maximizes the reuse of agent capabilities through multiple application domains, and supports the automatic, on-demand configuration of agent teams according to stated problem requirements. Considerable effort has been devoted to the applicability of the framework, which resulted in the implementation of an infrastructure to develop and deploy cooperative MAS. This infrastructure explores the idea of configuring an electronic institution on-the-fly to fit the specific requirements of each problem to be solved by a group of agents.

Introduction

Open MAS require a new way of managing and integrating agent capabilities based on *middleware*: connectivity software that enables multiple processes running on one or more machines to interact across a network. When implemented in MAS, the middleware layer is usually provided by *middle agents* (Decker, Sycara, & Williamson 1997), such as *matchmakers* (Decker, Williamson, & Sycara 1996), *facilitators* (Erickson 1996; Genesereth & Ketchpel 1997) and *brokers*(Nodine, Bohrer, & Ngu 1999). Typically, the function of a middle agent is to pair requesters with providers that are suitable for them, a process called *matchmaking*.

Matchmaking is the process of verifying whether the specification of an agent capability "matches" the specification of a request to do something (a task to achieve): two specifications "match" if certain relation holds between them, usually a capability being able to achieve some task. Since matchmaking compares the specification of requests and advertisements, both providers and requesters must share a common language to describe them. This language is usually called an Agent Capability Description Language (ACDL). Semantic matchmaking, which is based on the use of shared ontologies to annotate agent capabilities (Guarino 1997), improves the matchmaking process and facilitates interoperation. However, the reuse of existing capabilities over new application domains is still difficult because capabilities are usually associated to a specific application domain.

Moreover, in addition to capability discovery through matchmaking, we want an ACDL to support other activities involved in MAS interoperation, namely: invocation, composition (team-design), team-formation and coordination.

- *Invocation*: an agent willing to invoke the capability provided by another agent must provide the input data in an appropriate format and using a shared interaction protocol.
- *Composition*: refers to the aggregation of several capabilities to achieve a global team goal. This process requires a combination of matchmaking, capability selection, and verification of whether the aggregated functionality satisfies the specification of the global goal. We call this process Team Design.
- *Team Formation*: is the process of allocating tasks to agents, according to the constrains determined during the Team Design process. Team members can be selected among several candidate agents, either in a distributed or centralized manner, and agents must agree upon the interaction protocols to coordinate during the cooperative activity.
- *Coordination*: team members must synchronize their actions so as to avoid deadlocks and effectively cooperate.

We want an ACDL to support both requesters and providers through all these activities. Our main goals are to extend matchmaking so as to maximize capability reuse, and to support the automatic composition of capabilities according to stated problem requirements. In a wide sense of the word, our focus is on the reuse issue, that we define as how to reuse an agent capability for different tasks, across several application domains, and interacting with other capabilities provided by different, probably heterogeneous agents. Therefore, in order to maximize the reuse of agent capabilities, we have explored the potential of the knowledge-modelling stance and the ideas brought about by the componential approach to software development. The result of our work is ORCAS a multi-layered framework for the design, development, and deployment of Cooperative MAS in open environments. But instead of going through the ORCAS framework layer by layer, this paper describes the cornerstones of the ORCAS framework transversally.

The aspects of the ORCAS framework we focus herein are the ORCAS model of the Cooperative Problem Solving

(CPS) process, and the ORCAS Agent Capability Description Language (ACDL).

Overview of the ORCAS framework

This section provides an overall view of the framework and reviews the most important concepts needed to understand the other sections. The reader is referred to (Gómez 2004) for a complete, detailed description of the framework. ORCAS has three layers, namely: the Knowledge Modelling Framework, the Operational Framework and the Institutional Framework:

1. The *Knowledge Modelling Framework* (KMF) proposes a conceptual and architectural description of problem-solving systems from a knowledge-level view, abstracting the specification of components from implementation details. In addition, the KMF incorporates a bottom-up design process to configure a system out of elementary components, until a system configuration is found that satisfies some global requirements. This process is used to design the organization of a team and the competence required for each team role during the problem-solving process.
2. The *Operational Framework* deals with the link between the specification of components in the KMF, and the operational aspects of Multi-Agent Systems. This framework comprehends an extension of the KMF to obtain a full-fledged Agent Capability Description Language, together with a new model of the Cooperative Problem Solving process that includes a Team Design stage prior to the Team Formation stage.
3. The *Institutional Framework* describes an implemented infrastructure for developing and deploying Multi-Agent Systems configurable on-demand, according to the the requirements of the problem at hand.

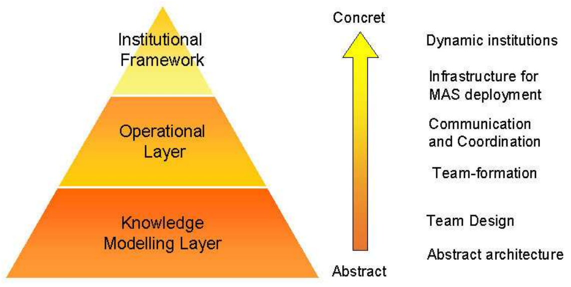

Figure 1: The three layers of the ORCAS framework

Figure 1 shows the three layers as a pyramid made of three blocks. The block at the bottom corresponds to the more abstract layer, while upper blocks corresponds to increasingly implementation dependent layers. Therefore, developers and system engineers can decide to use only a portion of the framework, starting from the bottom, and modifying or changing the other frameworks according to its preferences and needs.

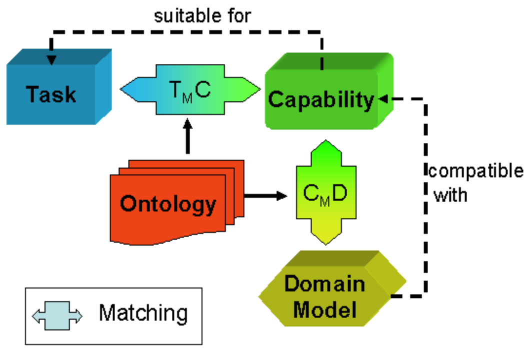

Figure 2: The ORCAS Abstract Architecture

Knowledge Modelling Framework

The ORCAS Knowledge Modelling Framework (KMF) proposes a conceptual description of Multi-Agent Systems at the *knowledge level* (Newell 1982), abstracting the specification of components from implementation details. The purpose of the *Knowledge Modelling Framework* (KMF) is twofold: on the one hand, the KMF is a conceptual tool to guide developers in the analysis and design of Multi-Agent Systems in a way that maximizes capability reuse across different domains; on the other hand, the KMF provides the basis for an Agent Capability Description Language (ACDL) supporting the automatic, on-demand configuration of agent teams according to stated problem requirements.

The ORCAS KMF is based on the *Task-Method-Domain* (TDM) paradigm prevailing in existing Knowledge Modelling frameworks. This paradigm distinguishes between three classes of components: *tasks*, *problem-solving methods* (PSM) and *domain models*. In ORCAS there are tasks and domain models, while PSMs are replaced by agent capabilities, playing the same role as PSMs, but including agent specific features concerning communication and coordination. Adopting this KMF we expect the ORCAS Abstract Architecture to provide an effective organization for constructing libraries with large "horizontal cover" (Breuker & Van de Velde 1994; Valente, Van de Velde, & Breuker 1994; Motta 1999), thus maximizing reusability and avoiding the brittleness of monolithic libraries (Motta *et al.* 1999).

A *task* is a functional description of a type of problem to be solved. A task is functionally characterized by *input roles*, *output roles*, and the relationship between them, which is specified as a set of *preconditions* and *postconditions*.

A *capability* describes a particular method for solving problems with some specific properties. A capability is specified from a functional viewpoint by stating the *input roles*, *output roles*, *preconditions* and *posconditions*. In addition, a capability can specify the type of domain knowledge (*knowledge-roles*) it requires, and some properties that have to be fulfilled by the domain knowledge to sensibly apply that capability (*assumptions*).

There are two types of capability: *task-decomposer* and *skill*. While skills are used to describe primitive, atomic reasoning steps, task-decomposers are used to describe complex reasoning methods that decompose a problem into several subtasks.

Finally, a *domain model* (DM) specifies the concepts, relations and properties characterizing the knowledge from certain application domain.

The ORCAS KMF extends matchmaking in two ways (Gómez & Plaza 2004): first, in addition to provide its own version of a *task-capability matching*, ORCAS introduces a *capability-domain matching* to decide whether a capability is suitable to be applied over certain application domain; and second, ORCAS addresses the composition of capabilities on-demand, according to the requirements of the problem at hand.

We regard the composition of capabilities as a "bottom-up design problem"(Hafedh Mili & Mili 1995), which in agent terms would be rewritten as: *given a set of requirements, find a set of agent capabilities whose combined competence and available knowledge satisfies those requirements*. The main difficulty to solve that problem is how to decompose the requirements in such a way as to yield component specifications. Our approach to this problem is to use a search process over the space of possible configurations. In ORCAS, the bottom-up design process is called Team Design, and the result is a *task-configuration*, a hierarchical decomposition of a task into subtasks, and capabilities bound to tasks according to matching relations, in such a way that the resulting task-configuration satisfies the global problem requirements.

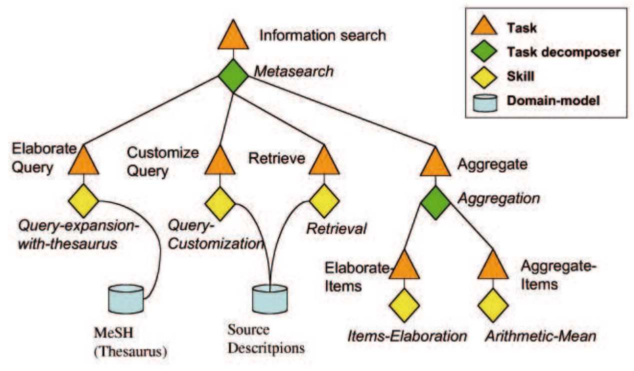

Figure 3: Task-configuration example

Figure 3 shows an example of a task-configuration for the Information-Search task, which is used within the WIM application. This task is being 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 in two subtasks: Elaborate-items and Aggregate-items. The example shows some skills requiring domain knowledge, e.g. 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.

Operational Framework

The Operational Framework describes a mapping from concepts in the Knowledge-Modelling Framework to concepts from Multi-Agent Systems. Specifically, the Operational Framework describes how a composition of capabilities represented at the knowledge-level can be operationalized by a customized team of agents; in other words, how to form a team of agents able to carry on the execution of a particular composition of capabilities, over a particular application domain.

The Operational Framework proposes a hierarchical model of teamwork that is straightforwardly derived from the hierarchical decomposition of tasks into subtasks that is the backbone of the TDM paradigm. This model of teamwork is embedded within a complete model of the Cooperative Problem Solving process that covers all the stages, from the specification of a problem to be solved to the activities carried on by agents willing to solve it.

In order to effectively use a KMF in open agent environments, a capability description language should include some way of specifying the communication and the coordination mechanisms required by agents to cooperate. Our approach to describe such aspects of a capability is based on the macro-level (societal) aspects of agent societies, which is focused on the communication and the observable behavior of agents, rather than adopting a micro-level (internal) view on individual agents. In particular, we are using concepts from the *electronic institutions* formalism to describe such aspects, as explained in the next section.

Institutional Framework

An electronic institution (e-Institution), is a "virtual place" designed to support and facilitate certain goals to the human and software agents concurring to that place (Noriega 1997; Rodríguez-Aguilar 1997). Since these goals are achieved by means of the interaction of agents, an e-institution provides the social mediation layer required to achieve a successful interaction: interaction protocols, shared ontologies, communication languages and social behavior rules.

The ORCAS Institutional Framework devises a institutional model covering all the stages of the ORCAS Cooperative Problem Solving process: the ORCAS e-Institution, a platform for developing and deploying cooperative MAS that supports both providers and requesters of capabilities along the different stages of the ORCAS model of the CPS process.

The e-Institutions formalism was originally conceived to deal with static organizations encompassing a fixed role structure and fixed interaction protocols, while the ORCAS institutional framework (both conceptually and in the implemented agent platform) is an electronic institution that allows other institutions to be configured on-the fly. The point is that in the ORCAS platform, each team formed to solve a problem implies that a new electronic institution is composed on demand out of the interaction protocols agents are equipped with, and that institution is used as the social mediation layer required by team members to effectively communicate and coordinate during the CPS activity.

The integration of the knowledge-modelling stance and the electronic institutions formalism within the ORCAS e-Institution favors the development of highly configurable and reusable MAS in open environments.

Remark that in addition to implement an agent infrastructure using the electronic institutions formalism, we are using the concepts proposed by the e-Institutions approach to specify the communication and coordination mechanisms of individual agents.

In order to demonstrate the applicability of the ORCAS framework, we have built WIM, a MAS-based configurable application to search bibliographic information in the Internet (Gómez & Abasolo 2003; Gómez 2004). WIM is an e-Institution that results of linking a library of information search and aggregation (ISA) capabilities provided by agents, and domain knowledge from medicine, and more specifically, Evidence-Based Medicine (EBM). Figure 4 outlines the architecture of the WIM application. The ORCAS e-Institution provides the mediation service for agents to communicate and cooperate. The institution provides a yellow pages service (through a librarian agent) where problem solving agents register their capabilities. The capabilities in the library are link to some domain models characterizing the application domain (EBM). User requests to perform search tasks solve problem are received through a Personal Assistant agent that expresses them using the ISA ontology. This agent acts as mediator between the user and the institutional agents providing the services required to carry on the CPS process: to find a valid task-configuration (Team Design), to allocate tasks to agents (Team Formation), and to coordinate individual agent behaviors to achieve the global task cooperatively(Teamwork). The figure depicts also the existence of wrappers, ad-hoc elements that are used to agentify external information sources, making them accessible to agents.

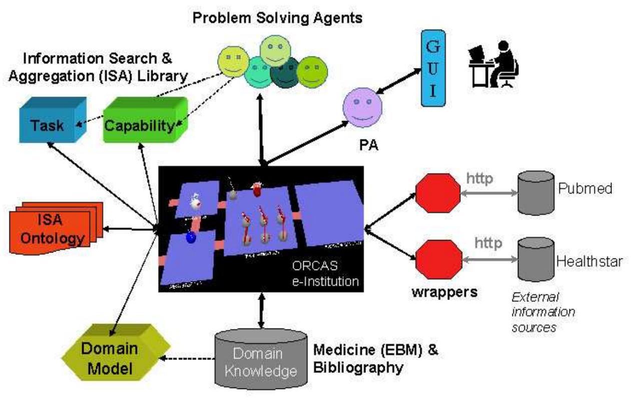

Figure 4: Task-configuration example

The ORCAS model of the Cooperative Problem-Solving process

Most of the research done in the field of cooperative MAS fits into one or more or the stages of the Cooperative Problem-Solving process as presented in (Wooldridge & Jennings 1994), which consists of four stages: *recognition*, *team formation*, *planning* and *execution*.

Although the proposers of this model believe many instances of the CPS process exhibit these stages in some form (either explicitly or implicitly), they stress that the model is idealized. In other words, there are cases that the model cannot account for (Wooldridge & Jennings 1999). Since team formation is not guided by a preplan to achieve the overall goal, but is just a commitment to joint action, then neither the agents joining a team (committing to carry on joint action) are guaranteed to play one role in the team once a plan was decided at the subsequent planning stage, nor the resulting team assures that a global plan can be found.

The SharedPlans theory (Grosz & Kraus 1996) and the frameworks based on it, e.g. (Giampapa & Sycara 2002), have emphasized the need for a common, high-level team model that allows agents to understand the requirements to achieve a global team goal and select a plan. Team plans are used by agents to acquire goals, to identify roles and to relate individual goals to team goals. A plan allows the initiator of the team formation process to know which are the subgoals and (optionally) the actions or capabilities required to achieve each subgoal. Therefore, the initiator of a CPS process can use an initial plan to guide the team formation process (Tidhar, Rao, & Sonenberg 1996). However, there is still another limitation of most planning frameworks used in real, implemented MAS: plans are tightly bound to a very specific domain and generic tasks. Consequently, plans are hardly reusable for different domains, and cannot be adapted to fit specific requirements of the problem at hand (e.g. different users may prefer different ways of achieving a task, and thus different capabilities may be preferable depending on the user preferences).

Finally, most planning-based MAS devise an internal perspective of agents, whose internal state is used as the basis for evaluating the cooperative behavior. Concerning this issue, though we recognize there are well founded reasons to adopt an internal perspective in several contexts, we think a external view has some notable advantages under certain circumstances; in particular, it is more appropriate for open systems because it avoids imposing a model of the internal agent architecture to external agent developers.

Summing up, we address some requirements for developing Cooperative MAS in open environments that are not entirely covered by previous models of the CPS process: (1) the need for initial plans to guide the team formation process; (2) the consideration of the user preferences and specific problem requirements as a constrain over the competence of a team; and (3) an external view centered on observable events, rather than an internal view imposing a particular agent architecture.

As a result of our work upon these issues we have conceived a new model of the CPS process with four subprocesses (Figure 5), namely Problem Specification, Team Design, Team Formation and Teamwork.

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

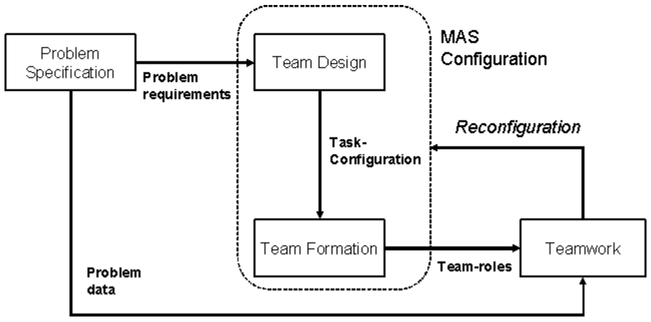

Figure 5: Overview of the ORCAS Cooperative Problem Solving process.

domain models) and the problem data to be used during the Teamwork process. The Team Design process uses the problem requirements to build a task-configuration —a composition of tasks, capabilities and domain models. The resulting task-configuration is used during the Team Formation process to allocate tasks and subtasks to agents, and instruct agents to ensure that the requirements of the problem are achievable. Finally, during the Teamwork process, team members try to solve the problem cooperatively, following the instructions received during Team Formation, and thus complying with the specific requirements of the problem at hand .

The ORCAS model of the CPS process should not be understood as a fixed sequence of steps. Actually, we have implemented strategies that interleave Team Design and Team Formation with Teamwork, thus enabling distributed configuration, lazy configuration and dynamic reconfiguration of teams on runtime(Gómez 2004).

The ORCAS Agent Capability Description Language

The functional description of a capability as provided in the Knowledge-Modelling Framework enables the automated discovery and the composition of capabilities, without taking into account communication and coordination requirements. Nonetheless, the invocation of capabilities and the interoperation of multiple agents are not supported by the functional description of a capability. In order to deal with these activities, we have to include information concerning the *communication* requirements of an agent over the capabilities he provides, and the *operational description*(control and data flow) of tasks-decomposers . Figure 6 shows the main concepts specifiable in the ORCAS ACDL.

In ORCAS, both the communication and the operational description of a capability are described using concepts from the electronic institutions formalism (Esteva *et al.* 2001), which adopts an external view on agents. Specifically, ORCAS uses *scenes* to describe the communication requirements of an agent over a particular capability, and performative structures to specify the operational description of a task-decomposer. Since we are using those concepts in a new way, a brief review of the electronic institutions concepts used in ORCAS is pertinent now:

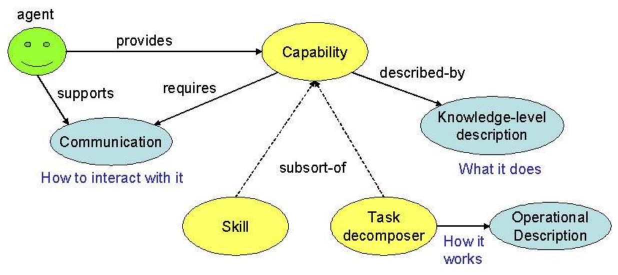

Figure 6: Main features of the ORCAS ACDL

1. *Agent roles:* Agents are the players in an electronic 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:* A dialogic framework determines the valid illocutions that can be exchanged among the agents participating in an electronic institution, which involves a vocabulary (ontology) and some agent communication language (ACL).
3. *Communication scenes:* A scene defines an interaction protocol among a set of agent roles, and using some dialogic framework.
4. *Performative structure:* A performative structure is 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.

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, then teamwork in ORCAS is straightforwardly organized as a hierarchy of team-roles. 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 can 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

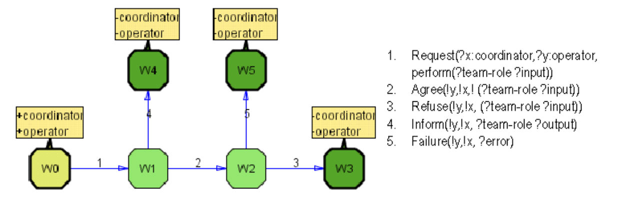
-coordinator
-operator
+coordinator
+operator
W0
W1
W2
W3
W4
W5
1
2
3
4
5
1. Request(?x:coordinator,?y:operator, perform(?team-role ?input))
2. Agree(!y,!x,! (?team-role ?input))
3. Refuse(!y,!x, (?team-role ?input))
4. Inform(!y,!x, ?team-role ?output)
5. Failure(!y,!x, ?error)

Figure 7: Example of a communication scene

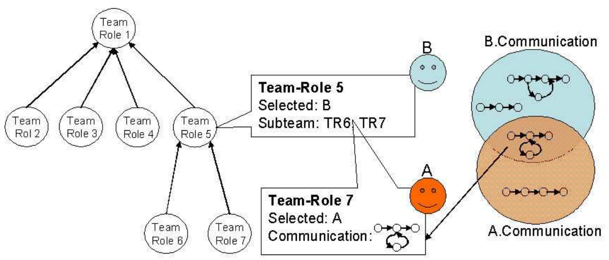

Figure 8: Choosing scenes during Team Formation

Figure 7 shows an example of a scene depicting the communication requirements of an agent over a capability. This example shows a typical request-inform protocol in terms of the ORCAS generic roles: *coordinator* (requester) and *operator* (informer).

Operational description

The ORCAS approach to specify the operational description of a task-decomposer is based on performative structures, with some distinctive features. In ORCAS, as in the e-institutions formalism, each scene within a performative structure corresponds to an interaction protocol. However, in ORCAS the scenes within a performative structure are not instantiated beforehand. Actually, these scenes are instantiated during the team-formation process using as a source the set of communication scenes shared by the agents willing to interact

Each scene 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. Both the coordinator and the operator must adhere to the same scene in order to communicate, and as a consequence, the scene must be chosen out of the scenes supported by both agents (see Figure 8). Since agents are selected dynamically during the Team Formation process, then the scenes used within ORCAS performative structures must also be chosen dynamically, before starting Teamwork.

Figure 9 shows an example of a performative structure specifying the operational description of the Aggregation task-decomposer, which decomposes a task into two subtasks: Elaborate-items and Aggregate-items. Therefore, this performative structure has two scenes (in addition to the Start and End scenes), one for each subtask, and three roles: x, y, z. There is one role of type *coordinator*(x) to be played by the agent applying the task-decomposer, and as many operators as subtasks. In the example there are two operators, one (y) participating in the Elaborate-items (EI) scene, and another (z) participating in the Aggregate-Items (AI) scene. Notice that the coordinator (x) is the same in both scenes, it enters first the EI scene, and can enter the AI scene only after finishing the EI scene.

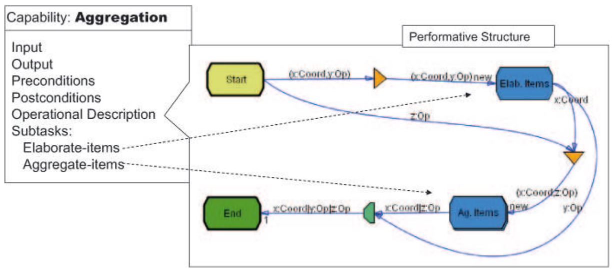

Figure 9: Task-decomposer operational description

So far, we have addressed the performative structure associated to a single task-decomposer, but what about the operational description of an entire team?

The ORCAS organization of a team adheres to the hierarchical decomposition of task into subtasks embodied by a task-configuration: starting from a top team-role, each team-role associated to a task-decomposer introduces a set of team-roles subordinated to it. Since each task-decomposer has an operational description described by a performative structure, therefore, the operational description of a team can be modelled as a nested structure of performative structures (Figure 10 shows an example). There is one performative structure for each task-decomposer, starting from the team-role associated to the root task of a task-configuration (the team-leader).

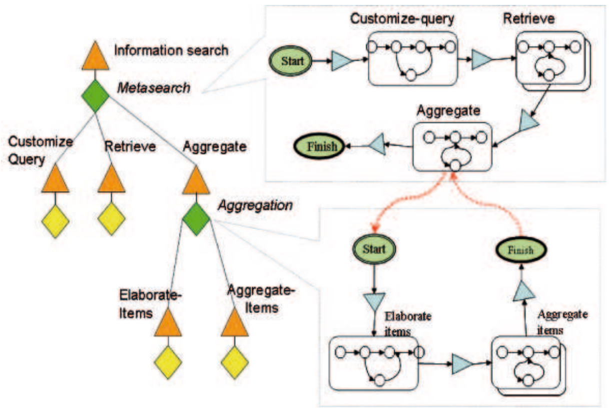

Figure 10: Teamwork as a nested structure of performative structures

Figure 10 sums up the specification of the teamwork activity as a nested structure of performative structures. Notice that there is a performative structure for each task-decomposer in the task-configuration, and there is one scene for each task. Performative structures embody which roles are required to play each scene, and the dependencies among scenes, e.g. some scenes must be finished before starting another scene, other scenes can be performed in parallel, and some scenes can be instantiated multiple times.

The teamwork process follows the hierarchical structure of a task-configuration, decomposing a task into subtasks when there is a task-decomposer. The teamwork process starts with a team-leader having to apply a task-decomposer. The team-leader initiates teamwork by following the performative structure specified in the operational description of its task-decomposer (Metasearch in Figure 10). Each scene within the performative structure refers to a communication scene to be played by the team-leader acting as coordinator and another agent allocated a subtasks, and playing the operator role. Since some of the subtasks may be bound to task-decomposers, a new performative structure must be carried over for each new task-decomposer. The first performative structure (the one initiated by the team-leader) cannot finish until the subsequent performative structures are finished, in other words, performative structures are nested.

In Figure 10, there are two task-decomposers (Metasearch and Aggregate), and thus there are two performative structures, one (Aggregate)) within the other. Each time a new team is formed complying with a task-configuration, a new structure of nested performative structures 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 capabilities of agents and the operational descriptions of tasks-decomposers.

Conclusions

The ORCAS framework explores the feasibility of the Problem Solving Methods (PSMs) approach to describe agent capabilities in a way that maximizes their reuse across multiple application domains. However, since PSMs were not designed having agents in mind, we have had to adapt them to deal with agent specific concepts such as communication, coordination and cooperation. Having situated our framework in the context of mediated architectures for cooperation, we have extended PSMs to obtain a full-fledged Agent Capability Description Language including not only the functional description of a capability, but also the communication requirements and the operational description of capabilities. We have adapted electronic institutions concepts to specify such aspects of a capability. Specifically, communication requirements are specified as scenes, and the operational description of task-decomposers is specified using performative scenes.

ORCAS integrates the architectural patterns that are the core of the knowledge modelling stance, and the requirements of an agent-centered approach to cooperative problem solving. The result is a model of the Cooperative Problem Solving process that enables a MAS to conform by to the requirements of every particular instance of a problem. The key of the model is a Team-Design stage, that applies and solves a ”bottom-up design problem” in the context of cooperative MAS. This process determines a possible configuration of a team in terms of the tasks to be solved, the capabilities required, and the domain knowledge available, in such a way that the specific requirements of the problem at hand are satisfied. ORCAS ideas are endorsed by the implementation of an agent infrastructure that has been tested in practice: the ORCAS e-Institution. But rather than being just an application of the e-Institutions formalism, the ORCAS infrastructure can be seen as a meta institution where dynamic problem-solving institutions are configured on-the-fly so as to satisfy stated problem requirements. Some elements from the electronic institution formalism have been adapted and incorporated as components of the ORCAS ACDL. These elements —-scenes and performative structures— are defined for each capability, and are used as building blocks to build a new electronic institution each time a team is formed. That institution configured on-the-fly is the shared social layer used by team members to effectively communicate and coordinate during the teamwork.

Some of our proposals are related to recent research on interoperability among heterogeneous agents. The way our approach describes agent capabilities is similar to the LARKS ACDL (**?**), used in the RETSINA infrastructure (**?**), but there also notable differences: first of all, ORCAS introduces domain models as a key element to maximize reuse of capabilities across several application domains; and second, while the RETSINA approach is focused just in the functional aspects of a capability (Grosz & Kraus 1996), the ORCAS framework covers also the communication and the coordination aspects of teamwork, using concepts from the e-Institution formalism.

We adhere to the view of Internet as an open environment where providers and requesters of capabilities meet and interact to solve specific problems by using the resources at hand. This view of Internet as a distributed computational platform is in spirit the same of the Semantic Web initiative, in particular, our view of agent capabilities is closer to Semantic Web Services frameworks such as the DAML-S ontology (The DAML-S Consortium 2001). From the Semantic Web Services paradigm, building an application is basically a process of composing, connecting and verifying the properties of web services in a way that resembles the compositional approach taken in the ORCAS Team Design stage. There are, however, two outstanding differences between Semantic Web Services and ORCAS. On the one hand, ORCAS agents are autonomous entities that can decide to accept or to refuse a request, while services are reactive, passive entities which are directly invoked by the client; therefore, instead of a centralized composition of services, we view the composition of capabilities as a negotiation process among autonomous agents. On the other hand, our language for describing capabilities is domain independent, since it is intended to maximize reuse, while existing SWS frameworks ignore this issue, since they are assume to be domain dependent by nature (a Web service is associated to some concrete domain, like the weather of a specific country in a weather forecasting service).

References

Bansal, S., and Vidal, J. M. 2003. Matchmaking of web services based on the DAML-S service model. In *Proceed-*

ings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems.

Breuker, J., and Van de Velde, W., eds. 1994. *CommonKADS Library for Expertise Modelling*. IOS Press.

Bryson, J. J.; Martin, D.; McIlraith, S.; and Stein, L. A. 2002. Agent-based composite services in daml-s: The behavior-oriented design of an intelligent semantic web. In Zhong, N.; Liu, J.; and Yao, Y., eds., *Web Intelligence*. Springer-Verlag.

Decker, K.; Sycara, K.; and Williamson, M. 1997. Middle-agents for the internet. In *Proceedings the 15th International Joint Conference on Artificial Intelligence*, 578–583.

Decker, K.; Williamson, M.; and Sycara, K. 1996. Matchmaking and brokering. In *Proceedings of the 2nd International Conference in Multi-Agent Systems*.

Erickson, T. 1996. An agent-based framework for interoperability. In Bradshaw, J. M., ed., *Software Agents*. AAAI Press.

Esteva, M.; Rodriguez, J. A.; Sierra, C.; Garcia, P.; and Arcos, J. L. 2001. On the formal specifications of electronic institutions. In *Agent-mediated Electronic commerce. The European AgentLink Perspective*, volume 1991 of *Lecture Notes in Artificial Intelligence*, 126–147.

Esteva, M. 1997. *Electronic Institutions: From Specification to Development*. Ph.D. Dissertation, Universitat Autnoma de Barcelona.

Genesereth, M. R., and Ketchpel, S. P. 1997. Software agents. *Communications of the ACM* 37(7).

Giampapa, J. A., and Sycara, K. 2002. Team-oriented agent coordination in the retsina multi-agent system. Technical Report CMU-RI-TR-02-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. Presented at AAMAS 2002 Workshop on Teamwork and Coalition Formation.

Gómez, M., and Abasolo, J. M. 2003. A general framework for meta-search based on query weighting and numerical aggregation operators. In *Intelligent Systems for Information Processing: From Representation to Applications*. Elsevier Science. 129–140.

Gómez, M., and Plaza, E. 2004. Extending matchmaking to maximize capability reuse (to appear). In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*.

Gómez, M. 2004. *ORCAS: Open, Reusable and Configurable Multiagent Systems*. Ph.D. Dissertation, Universitat Autnoma de Barcelona.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

Guarino, N. 1997. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In Pazienza, M., ed., *Summer School on Information Extraction*, 139–170. Springer Verlag.

Hafedh Mili, F. M., and Mili, A. 1995. Reusing software: Issues and research directions. *Software Engineering* 21(6):528–562.

Motta, E.; Fensel, D.; Gaspari, M.; and Benjamins, A. 1999. Specifications of knowledge components for reuse. In *Proceedings of SEKE '99, 1999.*

Motta, E. 1999. *Reusable Components for Knowledge Modelling*, volume 53 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Newell, A. 1982. The knowledge level. *Artificial Intelligence* 28(2):87–127.

Nodine, M.; Bohrer, W.; and Ngu, A. 1999. Semantic brokering over dynamic heterogeneous data sources in infosleuth. In *ICDE*, 358–365.

Noriega, P. 1997. *Agent-Mediated Auctions: The Fish-Market Metaphor*. Ph.D. Dissertation, Universitat Autnoma de Barcelona.

Park, J. Y.; Gennari, J. H.; and Musen, M. A. 1998. Mappings for reuse in knowledge-based systems. In *Proceedings 11th Workshop on Knowledge Acquisition, Modelling and Management*.

Payne, T. R.; Paolucci, M.; and Sycara, K. 2001. Advertising and matching daml-s service descriptions. In *Semantic Web Working Symposium (SWWS)*.

Rodríguez-Aguilar, J. A. 1997. *On the Design and Construction of Agent-mediated Electronic Institutions*. Ph.D. Dissertation, Universitat Autnoma de Barcelona.

The DAML-S Consortium. 2001. Daml-s: Semantic markup for web services. In *Proceedings of the International Semantic Web Workshop*.

Tidhar, G.; Rao, A.; and Sonenberg, E. 1996. Guided team selection. In *In Proceedings of the 2nd International Conference on Multi-agent Systems (ICMAS-96)*.

Valente, A.; Van de Velde, W.; and Breuker, J. 1994. The CommonKADS expertise modelling library. In Breuker, J., and Van de Velde, W., eds., *CommonKADS Library for Expertise Modeling*, volume 21 of *Frontiers in Artificial Intelligence and Applications*. IOS-Press. 31–56.

Wooldridge, M., and Jennings, N. R. 1994. Towards a theory of cooperative problem solving. In *Proceedings Modelling Autonomous Agents in a Multi-Agent World*, 15–26.

Wooldridge, M., and Jennings, N. R. 1999. The cooperative problem-solving process. *Journal of Logic and Computation* 9(4):563–592.