Integrating Knowledge Modeling and Multi-Agent Systems

Mario Gomez 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 matchmakers (Becker, Sycara, & Williamson 1997), such as matchmakers (Decker, Williamson, & Sycara 1996), facilitators (Erickson 1996; Genesereth & Ketchpel 1997) and brokers (Nodine, Bohrer, & Ng 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 interoperability. 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 constraints 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 goal is 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](image)

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 correspond 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.

**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 *postconditions*. 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 sev-
eral subtasks.

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

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...
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 composi-
tion of tasks, capabilities and domain models. The resulting
task-configuration is used during the Team Forma-
tion 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 un-
derstood as a fixed sequence of steps. Actually, we have im-
plemented strategies that interleave Team Design and Team
Formation with Teamwork, thus enabling distributed con-
figuration, 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 tak-
ing into account communication and coordination require-
ments. 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 capa-
bilities 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, OR-
CAS uses scenes to describe the communication require-
ments of an agent over a particular capability, and perfor-
mative 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 con-
cepts used in ORCAS is pertinent now:

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 re-
quired by agents playing part in given functional relations-
ships.
2. Dialogic framework: A dialogic framework determines
the valid illocutions that can be exchanged among the
agents participating in an electronic institution, which in-
volves 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 di-
alogic framework.
4. Performative structure: A performative structure is a net-
work of connected scenes that captures the relationships
among scenes. A performative structure constrains the
paths agents can traverse to move from one scene to an-
other, 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 capabil-
ity 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 de-
composition of tasks into subtasks, then teamwork in OR-
CAS 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 pro-
cessing between subtasks. In such an scenario, we can estab-
lish an abstract communication model with two basic roles:
coordinator, which is adopted by an agent willing to decom-
pose 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.