Chapter 1

Introduction

The main goal of this thesis is to provide a framework for open Multi-Agent Systems 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.

We have devoted considerable effort to the applicability of our proposals, which resulted in the implementation of an infrastructure to develop Multi Agent Systems according to the principles and requirements stated by our framework.

During the rest of this Chapter the main goal of this thesis is analyzed and boiled down to the several issues and problems it encompasses. First, we situate our work in the field of Multi-Agent Systems, focusing on the open problems and challenges that motivated us; second, the main contributions of this thesis are summarized; and third, the structure of the thesis is presented as a guide for readers.

1.1 Motivation and context

Distributed Artificial Intelligence has historically been divided in two main areas: Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS) [Bond and Gasser, 1988a]. In the DPS approach problems are divided and distributed among a number of nodes that cooperate in solving the different parts of the problem; but the overall problem solving strategy is an integral part of the system. In contrast, MAS research is concerned with the behavior of a collection of possibly pre-existing autonomous agents aiming at solving a given problem [Jennings et al., 1998]. MAS have been defined as loosely coupled networks of problem-solving entities working together to find answers to problems that are beyond the individual capabilities or knowledge of the isolated entities [Durfee and Lesser, 1989]. The MAS approach advocates decomposing problems in terms of autonomous agents that can engage in flexible, high level interactions, and this way of decomposing a problem aids the process of engineering complex systems [Jennings, 2000]. Some characteristics of MAS are the following:

- each agent has incomplete information or capabilities for solving the problem, thus each agent has a limited viewpoint;
- there is no global system control;
- data is decentralized; and
- computation is asynchronous.

Some reasons for the increasing interest in MAS research include: the ability to provide robustness and efficiency, the ability to allow inter-operation of existing legacy systems, and the ability to solve problems in which data, expertise, or control is distributed. Agents are defined as sophisticated computer programs that act autonomously on behalf of their users, across open and distributed environments, to solve a growing number of complex problems.

Considering the former definitions, MAS are supposed to be open systems in that agents can enter / leave at any time. Nonetheless, most of the initial work devoted to MAS research has focused on *closed* systems [Klein, 2000], typically designed by one team for one homogeneous environment, with participating agents sharing common high-level goals in a single domain. The communications languages and interaction protocols are typically in-house protocols, and are defined by the design team prior to any agent interactions. Systems are scalable under controlled conditions and design approaches tend to be ad hoc, inspired by the agent paradigm rather than using any specific methodologies.

It is often suggested the need for real open systems that were capable of dynamically adapting themselves to changing environments. Some examples are electronic markets, communities and distributed search engines. All in all, in open MAS the participants (both human and software agents) are unknown beforehand, can change over time and can be developed by different parties. Open systems are opposite to closed or proprietary systems, i.e. open systems can be supplied by hardware components from multiple vendors, and whose software can be operated from different platforms.

According to the predictions of the European Network of Excellence for Agent Based Computing, fully open MAS spanning multiple application domains and involving heterogeneous participants will not be achieved in a foreseeable future, and not before year 2009 [Luck et al., 2003]. This cautious prediction obeys to some challenges yet to be undertaken, including the following:

- provide effective agreed standards to allow open agent systems;
- provide semantic infrastructure for open agent communities;
- develop reasoning capabilities for agents in open environments;
- develop agent ability to understand user requirements;
- develop agent ability to adapt to changes in the environment;
- ensure agent confidence an trust in agents

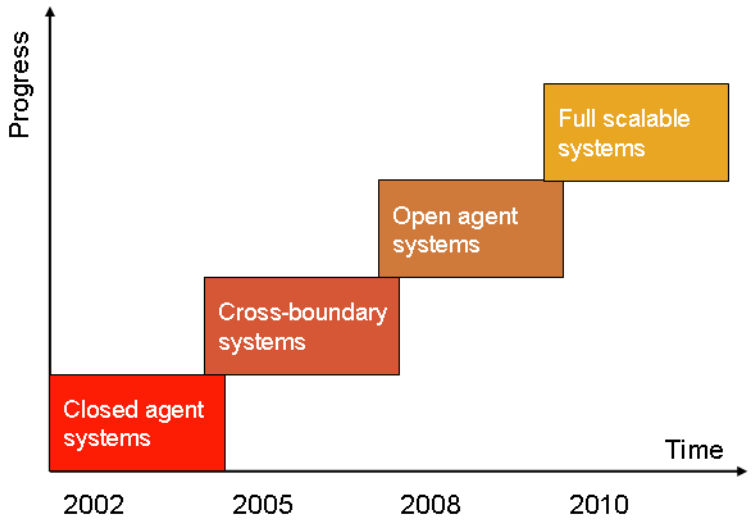

Figure 1.1: Roadmap timeline for agent technologies

Figure 1.1 (adapted from [Luck et al., 2003]) shows a roadmap timeline suggesting how agent technology will progress over time if R & D is aimed at the main challenges identified.

Although we are not going to solve all these problems entirely, we hope to provide tentative solutions to some of them and to bring about some insights that could drive future work on these issues. We are not going to exhaustively describe these problems here, since they are described in Chapter 2; we are rather going to sketch them so as to let the reader become acquainted with the motivations for this work.

Nowadays, MAS are increasingly being designed to cross corporate boundaries, so that the participating agents have fewer goals in common, although their interactions are still concerning a common domain. The languages and protocols used in these systems are being agreed and standardized; however, despite this raising diversity, all participating agents are designed by the same team designing the system and share common domain knowledge.

In order to overcome the limitations of current agent infrastructures for open MAS, researchers must tackle with several problems:

- the connection problem, or how to put providers and requesters (services and customers) in contact;

- the interoperability problem, or how to achieve a meaningful interaction among heterogeneous agents at the syntactic, semantic and pragmatic levels;

- the coalition problem, or how to form and coordinate agent teams to solve problems in a cooperative way;
- the reuse problem, or how to use the same agent capabilities across several application domains;
- the accountability problem, or how to predict or explain the behavior of MAS according to the requirements of the problem.

The MAS community builds intelligent agents capable of reasoning about how to cooperate to solve complex problems. This area uses knowledge intensively: for adding meaning (using ontologies), enabling service discovery and composition (using annotations and reasoning for matchmaking), and coordinating processes (using negotiation strategies). In closed environments knowledge is usually homogeneous and static. In open environments such as the Internet, knowledge is pervasive, distributed, heterogeneous, and dynamic in nature.

Nowadays the Web is shifting the nature of software development to a distributed *plug-and-play* process. This change requires a new way of managing and integrating software based on a software integration architectural pattern called *middleware*. Middleware is connectivity software; it consists of enabling services that allow multiple processes running on one or more machines to interact across a network. It follows that a middleware layer is required to provide a common set of programming interfaces that developers can use to create distributed systems.

Intelligent middleware aims to achieve the highest degree of interoperability, where systems can identify and react to the semantics of data. For this reason, many research communities are focusing their attention to semantic interoperability, for example: MAS, Semantic Web Services, Cooperative Information Systems and Component Based Software Development.

In open MAS the middleware layer is usually provided by *middle agents* [Decker et al., 1997b] that mediate between requesters and providers of capabilities, e.g. matchmakers [Decker et al., 1996], facilitators [Erickson, 1996a, Genesereth and Ketchpel, 1997] and brokers[Nodine et al., 1999]). Typically, the function of a middle agent is to pair requesters with providers that are suitable for them, and this process is called *matchmaking*. To enable matchmaking, both providers and requesters share a common language to describe the requests (tasks or goals) and the advertisements (capabilities or services) in order to compare them. This language is called an Agent Capability Description Language (ACDL).

Matchmaking is the process of verifying whether a capability specification matches the specification of a request (e.g. a task to be solved): two specifications match if their specifications verify some *matching* relation, where the matching relation is defined according to some criteria (e.g. a capability being able to solve a task). Semantic matchtmaking, which is based on the use of shared ontologies to annotate agent capabilities [Guarino, 1997a], improves the matchmaking process and facilitates interoperation.

Semantic matchmaking allows to verify whether a capability can solve a new type of problem (a task), but the reuse of existing capabilities over new

application domains is difficult because capabilities are usually associated to a specific application domain.

The notion of an *Agent Capability Description Language* (ACDL) has been introduced recently [Sycara et al., 1999a] as a key element to enable MAS interoperation in open environments. An ACDL is a shared language that allows heterogeneous agents to coordinate effectively across distributed networks. Sometimes, capabilities are referred as "services" and, consequently, an ACDL can alternatively be called an Agent Service Description Language (ASDL).

In the literature, an ACDL is defined as a language to describe both agent advertisements and requests, and is primarily used by middle agents (e.g. brokers and matchmakers) to pair service-requests with service-providing agents that meet the requirements of the request [Sycara et al., 1999b, Sycara et al., 1999a].

Some desirable features for such a language are *expressiveness*, *efficiency* and *ease of use*:

- *Expressiveness*: the language should be expressive enough to represent not only data and knowledge, but also the meaning of a capability. Agent capabilities should be described at an abstract rather than implementation dependent level.
- *Efficiency*: inferences on descriptions written in this language should be supported. Automatic reasoning and comparison on the descriptions should be both feasible and efficient.
- *Ease of use*: descriptions should not only be easy to read and understand, but also easy to write. The language should support the use of ontologies for annotate agent capabilities with shared semantic information.

However, in addition to capability discovery, an ACDL should bring support to other activities involved in MAS interoperation. On the one hand, once a capability is discovered, it should be enacted automatically; agents should be able to interpret the description of a capability to understand what input is necessary to execute a capability, what information will be returned, and which are the effects or postconditions that will hold after applying the capability. In addition, an agent must know the communication protocol, the communication language and the data format required by the provider of the capability in order to successfully communicate with it.

On the other hand, in order to achieve more complex tasks, capabilities may be combined or aggregated to achieve complex goals that existing capabilities cannot achieve in isolation. This process may require a combination of matchmaking, capability selection among alternative candidates, and verification of whether the aggregated functionality satisfies the specification of a high-level goal.

Our approach to these activities is tightly related with the idea of reuse: how to reuse a capability for different tasks, across several application domains, and in cooperation with other capabilities provided by different, probably heterogeneous agents. The idea of reuse is being addressed by the Software Engineering and the Knowledge Engineering communities.

The reuse of complete software developments and the process used to create them has the potential to significantly ease the process of software engineering by providing a source of verified software artifacts [Wegner, 1984]. It is suggested than reuse of software artifacts can be achieved through the utilization of software libraries [Atkinson, 1997]. Essentially a software library is a repository of information which can be used to construct software systems. The main goal of software libraries reuse is to enable previous development experiences to guide subsequent software development. To this end, MAS designers must be provided with libraries of:

- generic organisation models (e.g., hierarchical organisations, flat organisations);
- generic agent models (e.g., purely reactive agent models, deliberative BDI models);
- generic task models (e.g., diagnostic tasks, information filtering tasks, transactions);
- communication languages and patterns for agent societies;
- ontology patterns for agent requirements, agent models and organisation models;
- interaction protocol patterns between agents with special roles;
- reusable organisation structures; and
- reusable knowledge bases.

From the compositional approach, building a software system is essentially a design problem [Biggerstaff and Perlis, 1989]. The Component-Based Software development (CBSD) approach focuses on building large software systems by integrating previously-existing software components. By enhancing the flexibility and maintainability of systems, the ultimate goal is to reduce software development costs, assemble systems rapidly, and reduce the maintenance burden associated with the support and upgrade of large systems [Brown and Wallnau, 1996].

Constructing an application involves the use of prefabricated pieces, perhaps developed at different times, by different people and possibly with different purposes, therefore integrability of heterogeneous components is a key when considering whether to acquire, reuse, or build new components. Reusable software components can be deployed independently and are subject to composition by third parties [Szyperski, 1996]. There is, however, a major problem with software composition, the so called *Bottom Up Design Problem* [Mili et al., 1995], defined as:

> given a set of requirements, find a set of components within a software library whose combined behavior satisfies the requirements.

The fundamental difficulty when considering this problem is how to decompose the requirements in such a way as to yield component specifications. A reverse approach is to search the space of all possible component compositions until one satisfying the requirements is found [Hall, 1993, Zhang, 2000]. Thus, composition of components can be regarded as composition of their specifications [Butler and Duke, 1998].

Concerning Knowledge Engineering, we are interested in Knowledge Modelling Frameworks that has proposed several methodologies, architectures and languages for analyzing, describing and developing knowledge systems [Steels, 1990, McDermott, 1988, Schreiber et al., 1994a, Fensel et al., 1999]. The goal of a Knowledge Modelling Framework (KMF) is to provide a conceptual model of a system which describes the required knowledge and inferences at an implementation independent way. This approach is intended to support the engineer in the knowledge acquisition phase [Van de Velde, 1993] and to facilitate reuse [Fensel, 1997a].

However, KMFs and reusable software libraries have rarely been applied in the field of MAS to deal with the reuse and interoperation problems arising in open environments. This thesis explores the utility of a KMF to support the automated design and coordination of agent teams according to stated problem requirements; in other words, we translate the Bottom Up Design Problem problem to the MAS field: given a set of requirements, find a set of agent capabilities whose combined competence and knowledge satisfy the requirements.

1.2 Contributions

The main outcome of our efforts to overcome the problems concerning interoperability and reuse in open MAS is a multi-layered framework for MAS development and deployment that integrates Knowledge Modelling and Cooperative Multi-Agent Systems together. This framework is called ORCAS, which stands for Open, Reusable and Configurable multi-Agent Systems.

The ORCAS framework explores the use of a KMF for describing and composing agent capabilities with the aim of maximizing capability reuse and supporting the automatic, on-demand configuration of agent teams according to stated problem requirements. The ORCAS KMF is being used as an ACDL supporting semantic matchmaking and allowing capability descriptions in a domain independent manner, in order to maximize capability reuse.

The Knowledge Modelling Framework of ORCAS has been complemented with an Operational Framework, which describes a mapping from concepts in the Knowledge-Modelling Framework to concepts from Multi-Agent Systems and Cooperative Problem Solving. Specifically, the Operational Framework describes how a composition of capabilities represented at the knowledge-level can be operationalized by a customized team of problem solving agents. In order to do that, the Operational Framework extends the KMF to describe also 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. The reason to focus on the macro-level is to avoid imposing a specific agent architecture, thus facilitating the design and development of agents to third parties, a basic requirement of open MAS.

The ORCAS Operational Framework proposes a new model of the Cooperative Problem Solving process that is based on a knowledge-level [Newell, 1982] description of agent capabilities, using the ORCAS KMF. This model includes a Knowledge Configuration process that takes a specification of problem requirements as input and searches a composition of capabilities and knowledge satisfying those requirements. The result of the Knowledge Configuration process is a *task-configuration*, a knowledge-level design of an abstract agent team, in terms of the tasks to be solved, the capabilities to be applied, and the knowledge to be used by those capabilities.

An agent willing to start a cooperative activity requires an initial plan to know which are the capabilities required in order to select suitable agents for that plan. In larger systems, team selection may involve an exponential number of possible team combinations, and a blow-out in the number of interactions required to select the members of a team. There are two approaches to overcome these problems: one approach, that still relies on some kind of global plan is that of guiding the team formation with problem requirements [Tidhar et al., 1996]; another approach is to use distributed tasks allocation methods to make the team selection computationally tractable [Shehory and Kraus, 1998, Sandholm, 1993]; furthermore, a mixture of both approaches is also feasible [Clement and Durfee, 1999].

In this thesis we adopt the approach based on guiding the team formation process with the problem requirements, but the notion of a initial plan is here replaced by the notion of a task-configuration. A task-configuration reduces the complexity of the team formation process by constraining the composition of the team to a certain design that satisfies the requirements of the problem. In spite of its combinatorial nature, the complexity of the team selection process is mitigated, though partially transferred from the team formation process to the Knowledge Configuration process. Therefore, in order to further reduce the complexity of the Knowledge Configuration and the team formation activities, we propose Case-Based Reasoning to heuristically guide the search process over the space of possible configurations.

Finally, we have implemented and agent infrastructure according to the ORCAS model of the CPS process. This agent infrastructure has being implemented using the *electronic institutions* formalism [Esteva et al., 2001, Esteva et al., 2002b], which is based on a computational metaphor of human institutions from a macro-level point of view.

Human institutions are places where people meet to achieve some goals following specific procedures, e.g. auction houses, parliaments, stock exchange markets, etc. Intuitively, the notion of electronic institutions refers to a sort of virtual place where agents interact according to explicit conventions. The

institution is the responsible for defining the rules of the game, to enforce them and impose the penalties in case of violation.

An electronic institution, or e-Institution, is a "virtual place" designed to support and facilitate certain goals to the human and software agents concurring to that place. 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 interaction is not only regulated by the institution, furthermore it is mediated by institutional agents that offer an added value to participating agents.

The ORCAS e-Institution brings an added value to both requesters and providers: on the one hand, requesters are freed of finding adequate providers and provides a single interface to the multiple and heterogenous providers; on the other hand, the institution provides an advertisement service to capability providers, provides a mediation service for the team formation process, and facilitates coordination during the teamwork activity, allowing agents to solve complex problems that cannot be achieved by an agent alone.

However, in addition to implement an agent infrastructure using the electronic institutions formalism, we are interested on using the concepts proposed by the e-Institutions approach to describe the communication and the operational description of agent capabilities without imposing neither a specific agent architecture, nor an attitudinal theory of cooperation.

The goal of partitioning the ORCAS framework in layers is to bring developers an extra flexibility in adapting this framework to their own requirements, preferences and needs. We claim that a clear separation of layers will support a flexible utilization and extension of the framework to fit different needs, and to build different infrastructures. Therefore, we divide the ORCAS framework in three complementary frameworks:

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, a Knowledge Configuration model is presented as the process of finding configurations of components that fulfill stated problem requirements.

2. The *Operational Framework* deals with the link between the characterization of components and its implementation, that in our framework is realized by Multi-Agent Systems. This framework comprehends an extension of the KMF to become a full-fledged Agent Capability Description Language, together with a new model of the Cooperative Problem Solving process based on the KMF.

3. The *Institutional Framework* describes an implemented infrastructure for developing and deploying Multi-Agent Systems configurable on-demand, according to the the two layered —knowledge and operational— configuration framework. This infrastructure is designed and implemented ac-

cording to an institutional model of open agent societies. The result is multi-agent platform that supports flexible, extensible and configurable Multi-Agent Systems.

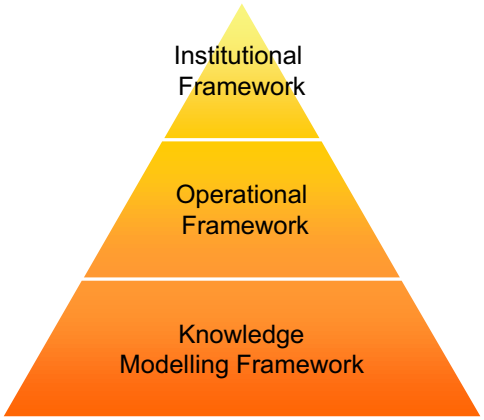

Figure 1.2: The three layers of the ORCAS framework

Figure 1.2 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.

1.3 Structure

This thesis consist of 7 chapters, including this one, and several appendixes providing technical information. The thesis is organized as follows (Figure 1.3):

Chapter 2 reviews some research relevant to our thesis and discusses some of their contributions that put the basis for our work, together with its limitations and the open issues we are dealing with. Since our work integrates two fields together -knowledge modelling and multi-agent systems-, this chapter have to address very different issues.

Chapter 3 draws the structure of ORCAS framework to give the reader an overall view of it, and remarks the outstanding elements of each layer so as to disclose the logic underpinning that structure.

Chapter 4 proposes a knowledge modelling framework for Multi-Agent Systems. This framework describes a conceptual and architectural characterization of problem-solving systems from a knowledge-level perspective,

abstracting the specification from any implementation details. Moreover, this chapter describes a Knowledge Configuration process that is able to find a configuration of components (tasks, capabilities and domain-models) fulfilling stated problem requirements.

Chapter 5 describes a framework to operationalize a knowledge-level configuration by forming and instructing a team of agents with the required capabilities and domain knowledge. This chapter describes also a model of teamwork based on the social view on agent cooperation.

Chapter 6 introduces the institutional framework, an implemented infrastructure for system development that is based on the two layered approach to multi-agent configuration together with an institutional approach to open agent societies, in support of flexible, customizable and extensible Cooperative Multi-Agent Systems.

Chapter 7 shows an implemented application as a case study of the ORCAS framework, the Web Information Mediator (WIM). WIM is an application to look for medical bibliography in Internet that relies on a library of tasks and agent capabilities for information search and aggregation, linked to a medical application domain.

Chapter 8 presents some conclusions and draws up those open issues that are believed to deceive future work.

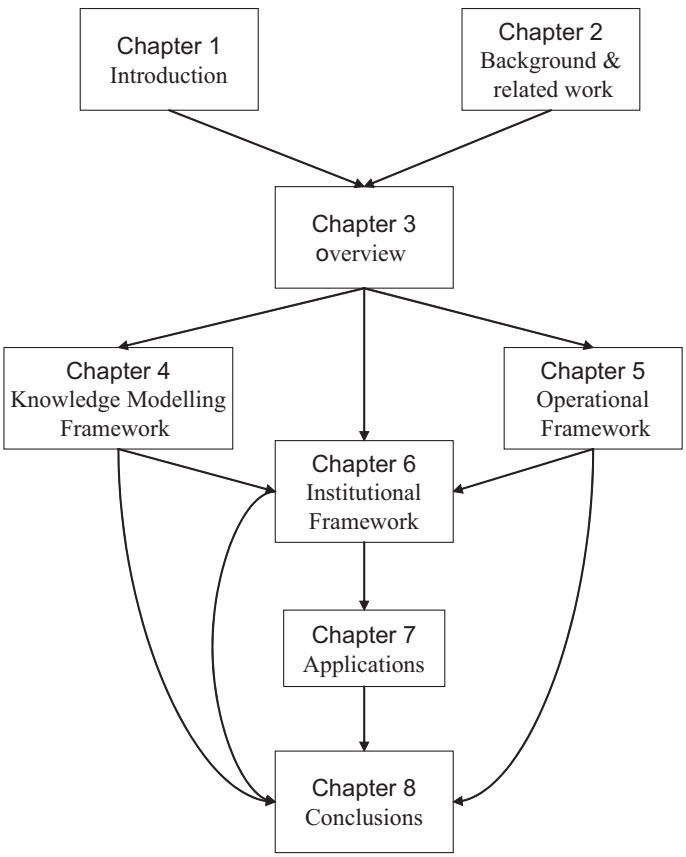

Figure 1.3: Thesis structure