

# A Flexible Multi-Agent Framework to Building Intelligent Systems

David Camacho, Fernando Fernández and Miguel A. Rodelgo  
dcamacho@ia.uc3m.es, ffernand@inf.uc3m.es, marodelgo1@telefonica.net  
Universidad Carlos III de Madrid  
Avda. de la Universidad 30, 28911-Leganés, Madrid, Spain

September 22, 2004

## Abstract

This paper describes an agent framework (SkeletonAgent) that has been designed and implemented to achieve two main goals. On the one hand, to design systems built by groups of agents that are able to integrate modules which implement Artificial Intelligence (AI) abilities. Those AI abilities, like planning or learning, are implemented as new agent skills and can be changed, or modified (in the design phase of the agent), to obtain new behaviours in the agents. On the other hand, this framework allows different types of agents, like deliberative agents or reactive agents, to be built in a flexible way, through the modification of certain agent architectural characteristics. The main goal of this paper is to demonstrate the flexibility of this framework. Therefore, this paper presents two different Multi-Agent instantiations: the first one is applied in the RoboSoccer Simulation League, and the second is used to allow the reutilization of Web information in a planning domain. Both systems are analyzed and experimentally tested to provide an accurate evaluation of this framework.

**keywords:** Multi-Agent Architectures, Robot Soccer Information Gathering, Web-based Systems, Distributed AI Systems.

## 1 Introduction

The Intelligent Agents and Multi-Agent Systems (MAS) research fields have experimented a growing interest from different research communities such as Artificial Intelligence (AI), Software Engineering, or Psychology. These research fields try to solve two distinct goals. On the one hand, they try to define and design software programs (usually called agents) which implement several characteristics such as autonomy, proactiveness, coordination, and language communication to obtain intelligent systems which are able to adapt the request provided to the inputs received from the environment [Jennings *et al.*1998]. On the other

hand, it is possible to coordinate several of these agents to build complex societies. When considering societies of agents, new issues arise, such as social organization, cooperation, knowledge representation, coordination, or negotiation. In this situation it is possible to speak about Multi-Agent Systems, and the previous problems can be studied within different perspectives [Brenner *et al.*1998]. There is a wide range of different domains that can be used to test agent organizations such as business management [Norman *et al.*1997], robotics [Brooks1991], the Web [Ambite *et al.*2002], or the simulation of complex societies [Dautenhahn1998]. A common point of interest for previous application domains is both how to define and design the individual agents that make up these systems [Petrie2000] and how to coordinate and organize groups of agents [Lesser1999].

The paper describes a generic Multi-Agent framework, called SkeletonAgent, that is capable of integrating AI classical techniques to build intelligent systems. This framework has characteristics learned from other agent-based models like CooperA [Sommaruga *et al.*1996], and from Multi-Agent models like *ABC*<sup>2</sup>, or Retsina [Decker and Sycara1997]. Our approach has extended previous models so that AI techniques (like Planning and Learning) can be applied in different domains (like Web Information Gathering and RoboSoccer) as new agent skills into a MAS.

Once the architecture is briefly described, the main goal of the paper is to deal with its instantiation into two heterogeneous domains to test its flexibility. The first domain is the problem of gathering information from the Web. The second domain consists of building a team of robot soccer agents, that can be used in the RoboCup simulation league [Kitano *et al.*1997]. The selection of these two domains is not casual, but they have been selected because the first domain requires deliberative agents that are able to cooperate and coordinate their capabilities find a solution, whereas the second domain uses reactive agents where decisions of the players are based mainly on immediate sensorial information. The systems have been implemented to work in both domains and will allow a demonstration of the flexibility of the SkeletonAgent architecture.

The paper is structured as follows: Section 2 describes the architecture of a generic software agent, and the related Multi-Agent framework (SkeletonAgent). Section 3 shows the first instantiation of SkeletonAgent into a MAS, called MAPWEB, which is applied in a travel domain. Section 4 provides another instantiation of SkeletonAgent in a reactive domain (RoboCup League). Finally, Section 5 summarizes the conclusions of the research.

## 2 SKELETONAGENT

In this section we briefly describe our software agent framework, where we address the main characteristics of the agent and Multi-Agent architecture defined by this framework.

## 2.1 Agent Architecture

Agents in SkeletonAgent are composed of several modules. Figure 1 shows those modules and their interconnections.

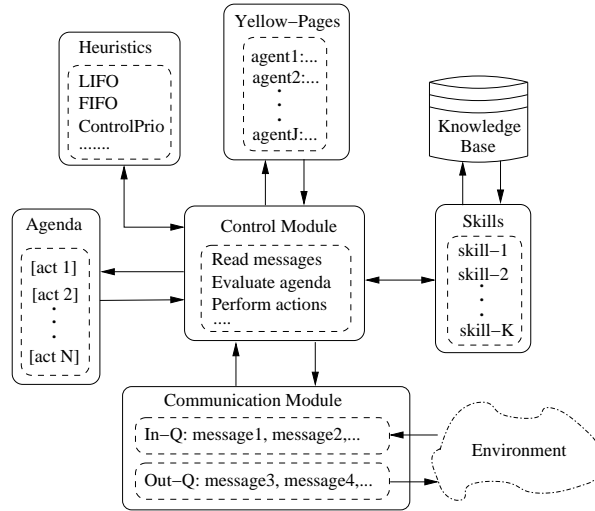


Figure 1: SkeletonAgent Architecture of an agent.

- Agenda/Skills** . The Agenda is a dynamic structure that stores items called *acts* which are directly related to a skill of the agent. These acts represent the actions that the agent is considering to select and carry out at a given moment. The content of these messages depends on the actual implementation domain. For instance, in the robosoccer domain, selecting the act “kick the ball” from the agenda will perform the agent skill (a routine) that carries out this act. These issues will be illustrated in Section 3.3.
- Heuristics/Control Module**. Agents have a set of heuristics that are used to decide at any time what act to select from the agenda. SkeletonAgent provides some standard behaviours like: *LIFO* (Last In/First Out), *FIFO* (First In/First Out), or a *ControlPrio* policy which allows the selection of the oldest acts, so that they can eventually be executed. These policies have to be selected before running the agent and remain fixed during execution time.
- Knowledge Base**. This module stores the knowledge that can be used by the agent skills. The architecture does not provide a language for representing generic knowledge. The programmer can use any structure that is required (files, databases, variables, etc). The actual knowledge stored depends on the domain. Examples for the e-tourism and robosoccer domains will be given in Sections 3.3 and 4.3.

- **Yellow Pages.** This module stores the knowledge that an agent has about all the agents belonging to its team. This information consists of a list made up of the name of its partners, and the name of the skills they can accomplish. Any skill can be considered as an abstraction of an action that will be accessible to other agents in the team. In fact, it means that the agent has meta-knowledge about itself (through its skills definition) and its partners.
- **Communication Module.** Agents in SkeletonAgent use a communication language to share information. It is the control module which decides to send a message and also the module which receives messages from other agents. Once the control module has received a message, it can be distributed to any other module of the agent. The communication process is implemented by three interconnected submodules: the *Communication Manager Module*, that manages the messages received from the Language Module in the input/output queue; the *Language Module*, that translates the internal data (used by the agent) into the communication language used by the agent (i.e. an standard KQML message) or vice versa; and finally, the *Communication Module* that is responsible for serializing and deserializing the information, so the message can be sent (received) through (from) the physical network.

## 2.2 Multi-Agent Architecture

In this section, how to implement societies of agents by means of the model outlined in the previous section is described. Any MAS defined using SkeletonAgent can be implemented using one or several teams. Every team is managed by a specialized agent named CoachAgent (*CCH*). It is necessary to use a single ManagerAgent (*MNG*) to manage the different teams. Therefore, in SkeletonAgent (like other MAS frameworks), any system implemented needs at least the following agents to work properly:

- *Control Agents.* They manage the different agents in the system. There are two types of them:
  - ManagerAgent (*MNG*). This agent is similar to *AMS* agents (as in FIPA), so its main roles are: to add and remove other agents from the system, and to control which agents are active in the agent society. This agent is responsible for building a team of agents. To do this, when any agent requests to be inserted in the society, the *MNG* determines which teams require this agent.
  - CoachAgent (*CCH*). They control a team of agents, guaranteeing stability and smooth cooperation of the active agents. The CoachAgents report problems directly to the *MNG* (for instance, when a new agent is required for the team). These agents are also used to guarantee that the yellow pages of the team members are coherent.

- *Execution Agents.* These agents are responsible for achieving the different goals of the system. To coordinate different teams of agents it is possible to include a new skill in the control module of the agents. Currently, there exist different kind of execution agents; we have implemented agents which are able to use a planner to solve problems (PlannerAgents), information agents that can retrieve Web data (WebAgents), or agents that can interact with the RoboSoccer simulator.

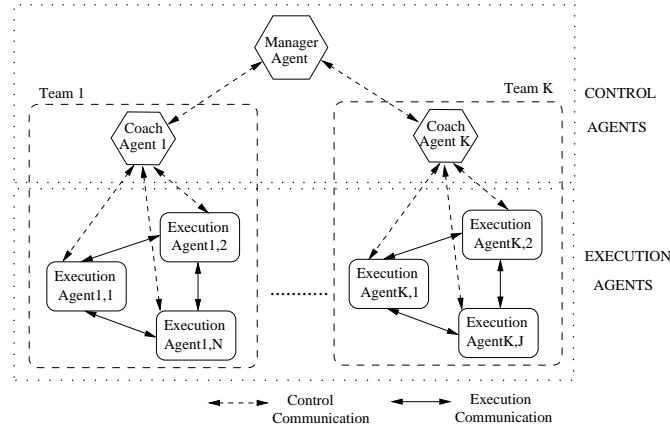


Figure 2: SkeletonAgent Multi-Agent Architecture.

Figure 2 shows the general architecture for any SkeletonAgent MAS. The main characteristics for any MAS implemented can be summarized in:

- Agents in the system use message-passing to communicate with other agents.
- All the agents have the same architecture and they are specialized in different tasks through the implementation of different skills.
- Although the communication language is the same for all the agents in SkeletonAgent, it is possible to distinguish two different types of communication messages. On the one hand, there are control messages whose main goal is to manage the behaviour of the system (control communication in Figure 2). On the other hand, execution messages are used to share knowledge and tasks among the agents, to achieve desired goals (execution communication).

To start the MAS correctly, it is necessary to perform the following steps:

1. First, the *MNG* is executed.
2. Agents in the system need to register themselves with the *MNG*. Once a *CCH* has registered, the *MNG* will select the necessary execution agents

from its white pages and will build an operative team. If there are not enough agents, the *CCH* will wait for them. To build a team the *MNG* selects the execution agents and provides the necessary information to the *CCH*. Once the information of the agents has been stored in *CCH*'s yellow pages, it updates the yellow pages of its execution agents. To select the necessary agents to build a group, the *MNG* uses the Ontology of the *CCH* agent.

3. Once a team is built, the execution agents can only communicate with the agents belonging to its team or with its *CCH*.

### 3 MAPWEB: Multi-Agent Planning in the Web

This section describes a specific instantiation of SkeletonAgent, named MAP-Web. MAPWeb is implemented by several software agents that allows accessing, extracting, and reusing information stored in several Web companies.

#### 3.1 E-tourism Domain

The high number of companies related to tourism, such as hotels, car rental, transport (flights, train, buses...), museums, theaters, etc. . . which have used the World Wide Web as a communication gateway between their resources and the potencial users has changed the concept of traditional tourism. Currently it is possible to consider that “*...travel and tourism represent the leading application in b2c (business to consumer) e-commerce... [Fodor et al.2002]*”.

The tourism domain is a complex and heterogeneous problem. It is possible to find heterogeneous information such as maps, textual information (about places to visit), schedule and fare information about hotels, transports, . . . It is possible for any user to consult (and even to buy) anything that s/he could need in a trip. All of this information needs to be appropriately managed and integrated to provide useful solutions. Different systems have been implemented to deal with the tourism information in the Web, such as Intelligent Electronic Tourist Guides (i.e. GUIDE [Cheverst *et al.*2000], Cyberguide [Abowd *et al.*1997]), Travel Assistants [Ambite *et al.*2002], Context-aware Systems for Tourism [Zipf2002]. The high relevance of this domain have interested in several research fields such as Information Technology [Werthner and Klein1999], or Artificial Intelligence (i.e. Case-Based Reasoning, Information Gathering, Information Retrieval,...) [Ambite *et al.*2002, Ricci and Werthner2002]. Although several AI techniques, such as Case-Based Reasoning (CBR) or Information Retrieval, have been successfully applied in this domain, we have defined the tourism domain as a planning domain, where the user needs to find a plan to travel between several places. This modification allows us to use clasical planning techniques both to manage the access to the Web companies that provide the information, and to look for heterogeneous solutions (integrating into a common solution the retrieved information from the Web). The main goal of this

domain is to allow the integration of AI techniques such as learning or planning with Information Gathering techniques into a real and complex domain.

Several Multi-Agent and Intelligent Systems have been implemented to work in the electronic tourism (e-tourism) domain [Staab and Werthner2002]. Some of them use mediation and mobile techniques to provide the solutions to the users as in CRUMPET [Schmidt-Belz], Information Gathering and Information agents as in the Travel Assistant built from Heracles and Theseus frameworks [Ambite *et al.*2002], or CBR techniques used to build Recommender Systems for tourism [Ricci and Werthner2002].

### 3.2 Multi-Agent characteristics of MAPWeb

MAPWEB [Camacho *et al.*2001a, Camacho *et al.*2005] (**M**ulti-**A**gent **P**lanning in the **W**eb) is a generic MAS information gathering architecture which has been implemented using the SkeletonAgent framework. MAPWEB is a Multi-Agent architecture that integrates planning and Web information gathering agents. This architecture provides a reusable code to help with the development of new Web gathering systems. The main goal of this framework is to deal with problems that require integrating planning with information gathered from the Web. MAPWEB implements a generic IG Web architecture which deals with the previous problems using Multi-Agent techniques to allow cooperation and coordination among heterogeneous agents (specialized in different tasks) and planning techniques to integrate information managed by those agents.

Figure 3 shows one possible MAPWEB topology, or configuration. This configuration is built by two operative teams managed by a *MNG*, and every team is locally managed by a *CCH*. *Team*<sub>1</sub> has the minimum set of agents to be operative, whereas *Team*<sub>2</sub> is built by *K* UserAgents, *P* PlannerAgents and *J* WebAgents. In addition, the following execution agents are needed:

- UserAgents are the bridge between the users and the system. They only implement basic input/output skills to acquire problem descriptions from users and to show the solutions found for them.
- PlannerAgents are able to solve planning problems using the information gathered from the Web.
- WebAgents are able to provide the requested Web information such as a set of relational records to the PlannerAgents using wrapping techniques. These agents use caching techniques to optimize the number of accesses to the Web [Camacho *et al.*2005].

Although MAPWEB is a generic architecture that combines Web information retrieval with planning, its skills are better understood in a particular domain. In this section, we will use the e-tourism domain, where the goal is to assist a user in planning his/her trips. MAPWEB's processes can be described as follows. First, the user interacts with the UserAgent to input

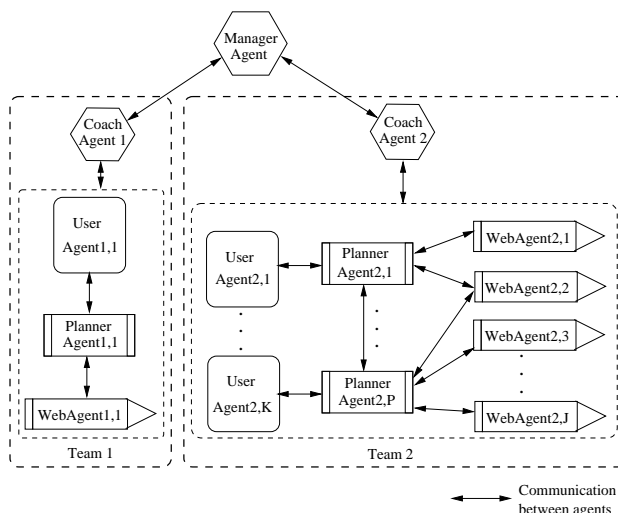


Figure 3: MAPWEB Architecture.

his/her query. The query captures information such as the departure and return dates and cities, one way or return trip, maximum number of transfers, and some preference criteria. This information is sent to the PlannerAgent, which transforms it into a planning problem. This planning problem retains only those parts that are essential for the planning process, which is termed the *abstract representation* of the user query. Then, the agent generates several abstract solutions for the user query. The planning steps in the abstract solutions require being completed and validated with updated information which is retrieved from the Web. To accomplish this, the PlannerAgent sends information queries to specialized WebAgents, which return several records for every information query. Then, the PlannerAgent integrates and validates the solutions and returns the data to the UserAgent, which in turn displays it to the user. These abstract solutions (skeletal plans) are used to represent the different steps that may be executed by the user. This process has been described in detail in [Camacho *et al.*2005, Camacho *et al.*2001a]. The skeletal solution obtained is used by the agent to achieve several goals:

- To coordinate other agents (user, planner, and Web agents). The planning agent uses the skeletal plan (abstract solution) as a template to decide what agent will be asked for help.
- To share the information stored in different agents (old plans in other planning agents, or records retrieved by Web agents).
- The skeletal plan is used as the integration structure to build the final solutions. The heterogeneous information provided by Web agents is integrated into specific solutions using the skeletal plan.

- To cooperate in the solving process with other planning agents. The planner agent can divide the initial problem into subproblems with less complexity (i.e. a problem with three independent goals can be divided into three new one-goal problems), and request help from other planning agents.

Currently, MAPWEB is able to access, gather, and reuse information from the following Web sources: Flight companies (Iberia Airlines, Avianca Airlines, Amadeus flights, 4Airlines flights), Train companies (Renfe, BritRail, RailEurope), Rental Car companies (Avis, Hertz, Amadeus car rental, 4Airlines car rental), and Hotel companies (Amadeus hotels, 4Airlines hotels). The previous Web sources can be classified into two main groups: Metasearch systems such as Amadeus, 4Airlines, BritRail, and RailEurope which extract information from several companies, and individual sources which belong to a particular company (Iberia, Avianca, Renfe, ...).

### 3.3 Characteristics of MAPWeb Agents

The previous section has briefly described the roles of control and execution agents used in MAPWEB. This section provides a description about the specific characteristics of these agents. Table 1 provides the specific characteristics of the control agents. Table 1 shows how it is only necessary to implement simple acts such as insert, or delete, in the *MNG* agent because the main role of this agent is to manage the correct insertion and deletion of the agents into the system. This agent uses a skill to distribute the agents into operative teams (it tries to implement teams with at least one operative agent). However, the number and type of agents that implement a particular team could change in time. For instance, a particular WebAgent could not be operative for network problems (the Web server is down), so the the *CCH* could suspend temporarily the agent, and resume again when this problem has disappear. The act:[Ask-for-agent] is used by the *CCH* agent to ask for a particular type of agent that is necessary for the proper functioning of the team. Finally, the KB (knowledge Base) and the Yellow Pages of those agents store all the related information with the managed agents.

Once the main characteristics of control agents have been described, Table 2 provides a description of the characteristics of the execution agents in MAPWEB.

As Table 2 shows the implemented UserAgent is the most simple execution agent, it only has a set of graphical interfaces to allow the communication between user and the system. The most remarkable issue in these agents is the implementation of a fuzzy skill which is used as a classifier to select and order the most promising solutions found (this algorithm uses some characteristics provided by the users in their profiles, or a predefined pattern which uses characteristics such as time travel or cost [Camacho *et al.*2001b]). The main goal of the PlannerAgents is to reason and deal with the Web information gathered by the WebAgents and build solutions for the given problems.

Table 1: Specific attributes for control agents.

	Manager Agent ( <i>MNG</i> )	Coach Agent ( <i>CCH</i> )
Agenda	The following acts are defined: <ul style="list-style-type: none"> <li>• Insert new agent</li> <li>• Delete agent</li> </ul>	The following acts are defined: <ul style="list-style-type: none"> <li>• Insert new agent in the team</li> <li>• Delete agent in the team</li> <li>• Suspend a particular agent</li> <li>• Resume a particular agent</li> <li>• Ask for new agent</li> </ul>
Heuristics/ Cont.Mod.	ControlPrio policy is used as predefined	ControlPrio
Skills	Is able to modify (in number and type) any team of agents. This agent uses a policy to distribute the new agents and to build the teams, or to share a particular agent in several teams (providing its direction to the CoachAgents of the teams)	They are able to manage a team, and to request for help to the <i>MNG</i> if any agent (PlannerAgent, WebAgent, User-Agent) is necessary to work correctly
Know. Base	Stores the information about all the agents in the system, naming information, skills, teams, etc. . .	Their KB stores the specific information about their execution agents and the contact address of the <i>MNG</i>
Yellow Pag.	[Name, direction, team, skills] of all the agents	[Name, direction, skills] of their related agents
Comm. Module	Is implemented using the KQML-ACL language and TCP/IP protocol	KQML-ACL language and TCP/IP protocol

For this reason, several techniques such as Planning and Case-Base Planning have been implemented, Likewise, the cooperation among PlannerAgents is allowed [Camacho *et al.*2001a]). Finally, the main characteristics in the WebAgents are the Automatic-Web-access and the Caching technique implemented. On the one hand, the automatic Web access is implemented through a set of specialized *Wrappers* that allows those agent to access, retrieve and store the information obtained from the Web sources in a standard format. On the other hand, the implementation of a Case Base that allows to those agents to minimize the number of access to the Web.

### 3.4 MAPWEB Validation

Once a system like MAPWEB is implemented it is possible to evaluate different characteristics such as the number of different travel plans found, the request time of the system, the scalability of the system when the number of agents/teams grown, the number of messages used to allow different processes such as coordination or cooperation among the agents, etc. . .

However, the goal of this paper is to show how a general agent framework can be used in heterogeneous domains. Therefore, this section only provides a brief evaluation of the behaviour of MAPWEB when a time limit is defined by the user. We consider that this experimental evaluation allows us to measure both the system performance to obtain useful solutions under time restrictions, and how the integration of AI techniques into deliberative agents affects solutions found.

Table 2: Specific attributes for execution agents.

	UserAgent	PlannerAgent	WebAgent
Agenda	These agents use acts that allow the user/system communication	These agents use acts that can be decomposed into more simple acts, when a PlannerAgent cannot be decomposed and act it tries to solve it	Simple acts that request information, and answer with the records found are used by these agents
Heuristics/ Cont.Mod.	FIFO	ControlPrio	FIFO
Skills	<p>The following skills have been implemented:</p> <ul style="list-style-type: none"> <li>• User/system communication. It allows to provide the problems, user characteristics (profile), and to show the solutions found</li> <li>• Fuzzy classification. A fuzzy algorithm is used to sort the solutions found by the system appropriately</li> </ul>	<p>The following skills have been implemented:</p> <ul style="list-style-type: none"> <li>• Case-Based Planning. It allows to store, retrieve, and retain old successful plans that can later be used by this agent (or by other PlannerAgents in the team)</li> <li>• Planning. It allows to use planners (we use Prodigy 4.0) as main reasoning module</li> <li>• Cooperation. It allows to ask other PlannerAgents for help in the solving process</li> </ul>	<p>The following skills have been implemented:</p> <ul style="list-style-type: none"> <li>• Automatic-Web-access. It allows to access, retrieve, and translate the data found in the Web sources into an relational format</li> <li>• A caching technique. It has been implemented using a database of retrieved records (only those records that finally are sent to the PlannerAgents are stored)</li> </ul>
Know. Base	Store information about the users (they can adapt the behaviour of the system defining his/her profiles), and information about the PlannerAgents that can help to solve a problem	The information to translate the user problem into a standard representation, the definition of the planning domain (e-tourism), and the Plan Base is stored, and managed by these agents	This module is built with the context information necessary to access and manage their related Web sources, and by a Case-Base of records retrieved from these sources
Yellow Pag.	Only the information about their related PlannerAgents is stored	All the contact information about their related UserAgents, PlannerAgents and WebAgents in the team	Only the information about their related PlannerAgents is stored
Comm. Module	KQML-ACL language and TCP/IP protocol	KQML-ACL language and TCP/IP protocol	KQML-ACL language and TCP/IP protocol

This experiment measures the number of plans (solutions) retrieved depending on the allowed time limit and the set of WebAgents used will be measured. To achieve this goal we took the following steps:

- Only international trips were considered, 30 planning problems were randomly generated.
- Then, all possible combinations of WebAgents were generated and tested by considering four specialized WebAgents: Amadeus-Flights (AMF, <http://www.amadeus.net>), 4airlines (4AL, <http://www.4airlines.com>), Iberia (IBE, <http://www.iberia.com>), and Avianca (AVI, <http://www.avianca.com>). Amadeus - Flights and 4airlines are metasearchers: they can search information from many airplane companies. Iberia and Avianca can only search information about their own flights.

- Finally, we have plotted the number of solutions retrieved depending on the allocated time limit for the combination that retrieves the largest number of plans.
- A simple configuration of MAPWEB, using one UserAgent, one PlannerAgent (so cooperation is not possible) and a set of WebAgents was implemented.

Table 3 shows the number of problems solved for the different MAPWEB topologies that use only an isolated WebAgent, and the problems solved for the best topology tested for the different possible WebAgents combinations. As is shown in Table 3 the best isolated WebAgents are the metasearcher engines (AMF and 4AL) given that these agents are able to retrieve information from different companies.

Table 3: Number of solved problems (out of 30) using different topologies.

Topology type	Selected topology	Problem type	
		0 Transfers	1 Transfer
1 WebAgent	AMF	15	26
	4AL	10	22
	IBE	7	15
	AVI	1	2
2 WebAgents	AMF-4AL	17	28
	AMF-IBE	18	29
3 WebAgents	AMF-4AL-IBE	18	29
4 WebAgents	AMF-4AL-IBE-AVI	19	30

Table 3 shows that using different specialized WebAgents is useful in solving more problems: the best single agent configuration (AMF) solves 15 problems of 0 transfers and 26 of 1 transfer, whereas the four agent configuration (AMF-4AL-IBE-AVI) manages to solve all the problems. However, adding more agents (3 and 4 agent topologies) is not always beneficial with respect to solving problems: a simple 2 agent configuration already fulfills 17/28 and 18/29 problems. But larger topologies usually find more solutions per problem, which can be useful if the user wants to choose a solution from a set of many possible plans according to some personal preferences (such as travel time, cost, etc.). This can be seen in Figure 4, which shows the average number of plans per problem found for each of the three categories described in the experimental setup. From these experimental results it is possible to extract the following conclusions:

- On the one hand, the number of problems solved for the AMF-4AL (17/28) and AMF-IBE (18/29) configurations showed in Table 3 provides an interesting result. Although, the first configuration uses two metasearch engines that can access and retrieve information from several companies (including the Web companies like Iberia or Avianca), the second ones (that uses the best metasearch engine and our best specialized WebAgent) obtain better results. Table 3 shows for the second configuration in both cases (0 and 1 transfers) that it is able to solve more problems. This

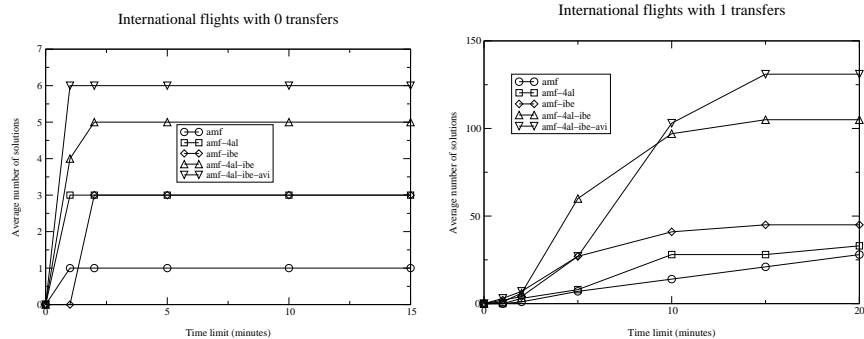


Figure 4: Average number of solutions for 0 and 1 transfers.

behaviour means that the specialized WebAgent (IBE) is able to provide new information that the metasearch engine (AMF) does not have. Therefore, the integration achieved by PlannerAgents through the utilization of abstract solutions allows us to find new solutions that are not possible to find without this integration.

- On the other hand, the combination of different techniques like Case-Base Planning (PlannerAgents) and Caching (WebAgents) allows us to increase the performance of the whole system using the old successful information stored in those agents. These techniques try to avoid several computational complex process, like the planning processes, or accessing to searching for information in the Web. Figure 4 shows the relation between the time limit defined by the user and the number of specific solutions found by the system. The system is able to find at least one useful (specific) solution in a few minutes (less than 3 minutes), searching in several Web sources and gathering data from several hundred of records.

## 4 ROBOSKELETON: Implementation of a Team of RoboSoccer Agents

This section describes a specific instantiation of SkeletonAgent in the RoboCup domain, which we have called ROBOSKELETON. It is a Multi-Agent team of software robots (reactive agents) that allows us to implement different control strategies for playing soccer. In this case, the control system is based on RECCA [Fernández *et al.*2000], a REactive Controlers based Cooperative Architecture.

The next subsection introduces the RoboCup domain, as well as the control algorithm used for the RoboSoccer agents. From that point on, the design of this multi-agent system, as well as the control algorithm, in the SkeletonAgent architecture is described in the following subsections.

## 4.1 RoboSoccer Domain

### 4.1.1 RoboCup and MAS

The Robot World Cup Initiative (RoboCup) [Kitano *et al.*1997] provides a standard problem, soccer, for the research of AI and intelligent robotics. Added to the real robot leagues, RoboCup provides a software platform, called Soccer Server simulator, for research and simulates several problems which appear with real robots. The Soccer Server is an environment for confronting two teams of players (software agents) [Noda1999] that are controlled by several types of systems. A match is carried out in a server-client style: a server, Soccer Server, provides a virtual field and simulates all movements of a ball and players. The clients become the players' brains and control their movements. Communication between the server and each client is done via UDP/IP sockets. Therefore, users can use any kind of programming system which has UDP/IP facilities. In another level of the communications, different protocols are developed between the clients and the server to transmit sensor information and agent control commands.

Given this platform, several architectures have been applied to control a robosoccer team. In [Matellán and Borrajo2001], an agenda based architecture is used to integrate the different skills of the players, each of them composed of different basic actions such as kick the ball, or send a message to another player. However, when integrating these actions in order to obtain complex behaviours, the designer must decide on a number of hierarchical levels in the control architecture, as defined in [Stone2000]. This decision depends on the knowledge about the model of the domain. If the model is very well known, it is possible to obtain a very detailed description of each task, so the domain, and how to solve the tasks in it, can be indepthly described and hierarchized. When the model is very dynamic, stochastic, and unknown, as in the case of RoboSoccer, the development of this hierarchy of tasks is not easy to create, given that the model of the domain must be created dynamically while interacting with it, and how to integrate pre-defined knowledge with the models being learned is not an easy task. Thus, reactive architectures are very robust in domains where a description of the model is not known, and are hard to build.

However, to coordinate reactive systems or reactive agents, a centralized reasoning is typically used, whic indicates to each agent the task that it has to solve. However, in domains like the Robosoccer, a centralized reasoning can not be implemented, because it requires a perfect knowledge of the domain, and a high communication capability that is not provided by the soccer server simulator. A simple way to control the team members and to obtain a collaborative behaviour among the agents can be achieved through the implementation of the REactive Controlers based Cooperative Architecture (RECCA) [Fernández *et al.*2000] which is described next.

### 4.1.2 RECCA

The main goal of this control architecture, designed for Robosoccer agents, is that the different players of the team are able to keep a play strategy like a 4-4-2 strategy (4 defenders, 4 midfield players, and 2 forwards) or any other strategy (4-3-3, 5-3-2, etc.), only by using local information. That means that the players will play without knowing the absolute locations of other players, so they will only use the visual information that they receive in each moment. So it is desirable that the players maintain the strategy whatever the situation of the game is, i.e. if they own the ball, if they are defending, etc. . . The architecture is based on two elements. In the first one, a play strategy is defined, so the team must follow that strategy, which can be modified by the coach of the team, but it is supposed that it will not change many times in a match. In the second one, each player plays a role in the previous strategy. This role is defined by three elements: the absolute position where the player can be located, the global leader and the local leader of each player. Figure 5 shows these concepts, for a team with a 4-3-3 strategy. The play area of each player is defined by the rectangles, and the local leaders are defined by arrows. The global leader (which owns the ball) is the only player who does not has a local leader.

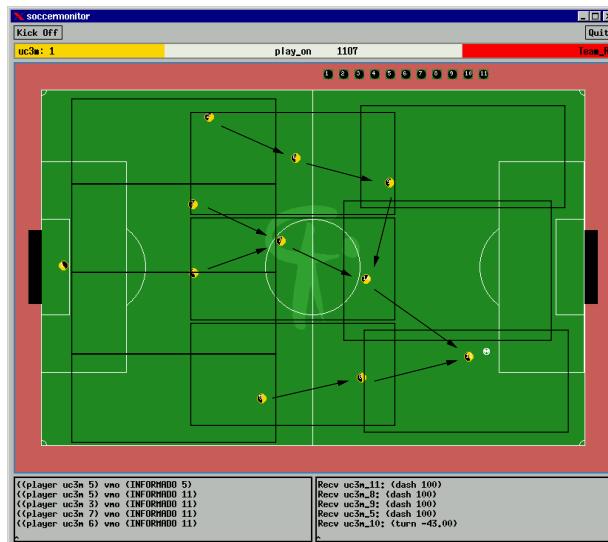


Figure 5: Local areas and leaders.

The play area associated to each player is defined “a priori”, and depends on the strategy. These areas must be big enough to allow the players to cover the field and to follow their local leader, but small enough to maintain the coherence of the team. The global leader is the player that owns the ball, or that is supposed to be able to obtain it, and it is the only global information that all the agents must share. The global leader of the team changes in time,

and is the center of the play for its team. So, if one player is the global leader, all the other players will play using it as the central point of the team. The decision about who the global leader is, is defined by a reduced communication protocol. If a player is very close to the ball but it is not the global leader, it can send a message to the rest of the team members informing that he is the new leader, obtaining the ball. Furthermore, a global leader may give the leadership to another team member when, for instance, he passes the ball.

The local goal of each player is to follow his local leader. The idea is that if the players are able to follow their local leader, they will keep the formation defined by the play strategy. Local leader of each player depends on the global leader, so depending on who the global leader is, the local leader of each player may be different. This relationship among global and local leader is defined with the play strategy and the formation of the team.

The control algorithm of all the players (except the goal keeper, which is implemented differently) is defined in Figure 6. It shows that all the information used to take the decisions is local to each player, except for the one who is the global leader which, as defined above, is the only information shared by all the team. Each of the actions shown in the flow represent high level acts that can be executed by the agents. For instance, the first action that all the players do is to verify if they are blocked by another player, i.e. if another player is in its way, and to unblock the situation. There are several different acts defined. For instance “look for the ball” is an act where the player rotates to find the ball. Acts like “go to the ball”, “get closer to my local leader”, or “go to the center of my play area” makes the agent move to different locations. “Change global leader” and “Look for the local leader” are the only two communication acts. In the first one, the new global leader is communicated to the rest of the players. In the second one, a player ask for the position of its local leader, so its local leader, or any other player who knows his position, may answer this position.

The control scheme has two main branches. The first one, if the player is the global leader, where it tries to go forward or pass the ball to a partner. The second one, if the player is not the local leader, where it tries to stay in its play area, following to its local leader. Each of the actions shown in the flow could be studied to improve the behaviour of the team so, for instance, if a player is not the global leader, instead of keeping in the center of its play area, he could for search a good position to receive the ball, etc.

## 4.2 Multi-Agent characteristics of RoboSkeleton

The general multi-agent architecture that supports SkeletonAgent was introduced in Figure 2, showing three main kinds of agents, manager agents, coach agents and execution agents. Figure 7 shows how this architecture has been adapted to the RoboSoccer domain.

Manager Agent appears to control the number of agents in the system, taking into account the restrictions of this domain: only two teams, each of them composed of 11 players. In the case of implementing robot soccer teams in the Soccer Server Simulator, this manager agent is implemented by the simulator

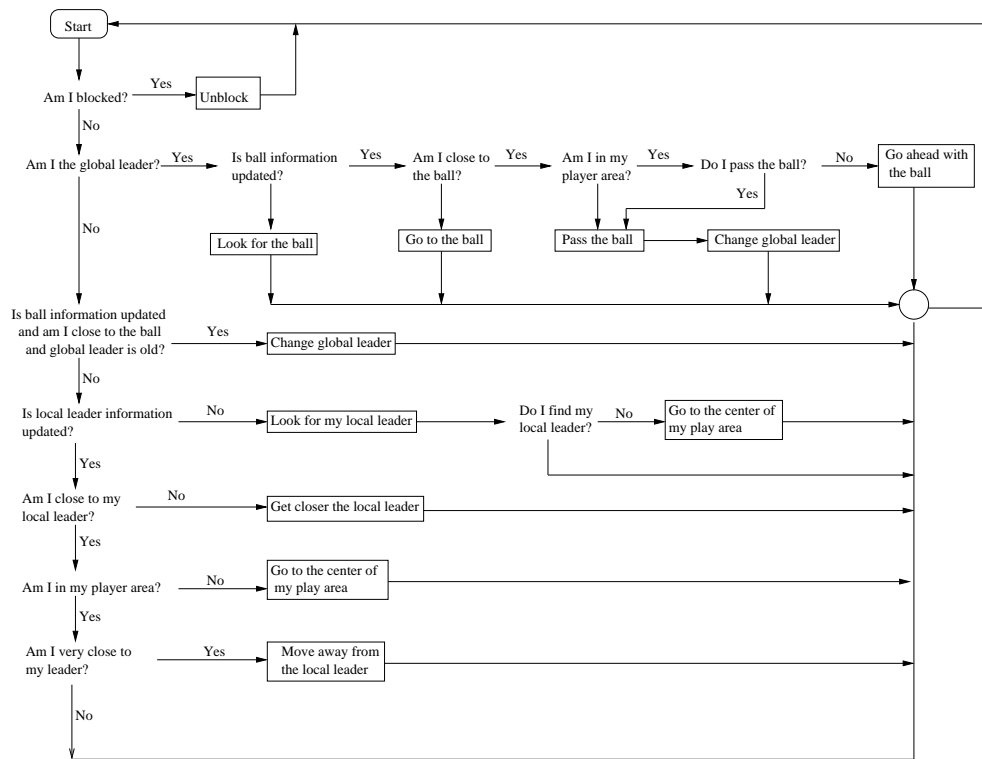


Figure 6: RECCA control algorithm for players.

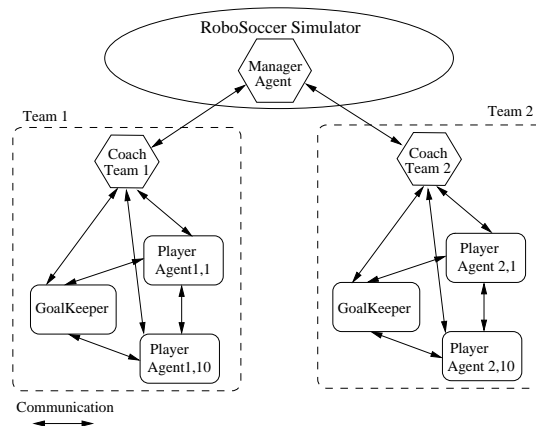


Figure 7: RoboSkeleton Multi-Agent Architecture.

itself, which is the element to which the different agents of the different teams must connect. Nevertheless, Soccer Server simulator implements all the roles defined for this kind of agent in section 2.2, i.e. to add and remove other agents from the system (1 coach and 11 players per team), control which agents are active in the agent society and group agents in teams.

A coach agent manages the agents of a team, guaranteeing stability and smooth operations of the active agents, as defined in section 2.2. Therefore, in the RoboSoccer domain, this agent must define the play strategy of the team, the role that each player must play, etc. . . . In the robot soccer domain, coaching is demonstrating a very important impact in the results obtained by the teams [Riley *et al.*2002]. We suppose that our coach has three main capabilities, following the ideas introduced in [Riley and Veloso2001]. The first one is modelling opponent behaviour in order to characterize it from a predefined set of different possible opponent models. The second one, and once opponent behaviour has been characterized, is deciding the right play strategy. And lastly, is communicating the new strategy to the players. This last capability is the only one implemented up to now in our software, but we will show how the architecture should work with the other ones.

Execution agents are responsible for achieving the different goals of the system, that in in the RoboSoccer domain is defined as winning the match. In this case, RECCA defines two different kind of agents, the goalkeeper and the rest of the players, i.e. defender, midfield and forward players. The role of each player depends on the coach, so it does not need to be defined at this level.

### 4.3 Characteristics of RoboSkeleton Agents

This section provides a detailed description about the specific characteristics of each agent in the ROBOSKELETON system. The Table 4 describes the specific agent characteristics.

- **Control Agent: ManagerAgent** (*MNG*). As we have stated before, in the Robocup application of the multi-agent SkeletonAgent architecture, this agent is implemented by the Soccer Server Simulator, so it has not been implemented following the standard agent architecture.
- **Control Agent: CoachAgent** (*CCH*). The acts that this agent manages are related to defining and communicating the strategy to the players. This communication can be defined at the beginning of a match, but acts for identifying the strategy of the opponent, and from that model, defining a new strategy, are included. The knowledge base introduces two kind of information. On the one hand, the information required to model the behaviour of the opponent. On the other hand, the information required to define the play strategy from the opponent model. Players are very homogeneous, so no yellow pages are required, given that the rest of agents will be treated as equal. Communication module follows the ideas of SkeletonAgent, establishing the three communication levels.

Table 4: Specific attributes in the ROBOSKELETON agents.

	Manager Agent	Coach Agent	Goal Keeper	Player
Agenda	-	The following acts are defined: <ul style="list-style-type: none"> <li>• Communicate the strategy to the players</li> <li>• Extract opponent model</li> <li>• Define new play strategies from opponent models</li> </ul>	The following acts are defined: <ul style="list-style-type: none"> <li>• Look for the ball</li> <li>• Locate in goal</li> <li>• Catch the ball</li> </ul>	The following acts are defined: <ul style="list-style-type: none"> <li>• Look for the ball</li> <li>• Look for the leader</li> <li>• Change leader</li> <li>• Others</li> </ul>
Heuristics/Cont.Mod.	-	FIFO/sequential	FIFO	RECCA
Skills	-	It can communicate with the player agents to inform about changes in play parametres, like play strategy, size of the play areas, etc.	Low level skills to catch the ball and to kick it to another team member	All the skills related with playing soccer, as well as capabilities to connect to the other agents
Know. Base	-	Opponent models, play strategies, information relationing opponent models and successful play strategies	-	Play strategies, global and local leader information
Yellow Pag.	-	-	-	-
Comm. Module	-	Soccer Simulator Language and UDP/IP protocol	Soccer Simulator Language and UDP/IP protocol	Soccer Simulator Language and UDP/IP protocol

- **Execution Agent: GoalKeeper.** The goal keeper has an independent implementation from the rest of the players. His only behaviour is to keep in his goal, and to move only when a forward player kicks the ball to the goal. This last action can be hand-made programmed or learned with machine learning methods, as defined in [Fernández and Borrajo2000]. Knowledge managed is limited to sensorial information.
- **Execution Agent: Player.**

There are three kinds of skills, that compose a hierarchy of skills. The first one is a basic skill that corresponds with an action executable in the simulator, like the *turn* and *move* soccer server commands. The second one is a single skill, that is composed of a sequence of basic skills, but that does not require external information to be executed, for instance, a sequence of *dash* commands. The third level is composed of skills that can involve complex behaviours, and can be composed of basic and single skills, and decisions can be taken depending on additional information of the agent, the server, etc, that can be received while the skill is being executed. The last two kinds of skills are the ones defined in the RECCA

control architecture shown in Figure 6.

Even though the players follow a reactive behaviour, some information is managed: On the one hand, information computed from the information received through the sensors; On the other hand, information required to execute the RECCA control scheme, such as the size of the play area and the relationship among global and local leaders, that are given by the play strategy. Furthermore, global leader is the only knowledge shared by the team, and it requires a communication process among the agents, as defined above. Players are homogeneous, so no yellow pages are required.

Communication among the agents can follow the three layers architecture defined for SkeletonAgent in Section 2.1. So, the UDP/IP protocol is used in the Communication Layer, the Soccer Server communication language is used in the Language Layer, and each agent has its own Communication Manager Module, defined to process all the messages received and that it needs to send to other players. Furthermore, with the Soccer Server Simulator, all communications are centralized by the server because any other kind of communication is forbidden by RoboCup simulation league rules.

#### 4.4 ROBOSKELETON Validation

To evaluate the RoboCup team implemented we have considered several factors that can be taken into account, both from the quantitative (number of goals scored, lost balls, successful passes, etc), and the qualitative point of view (individual and team coordination, play strategy execution, etc.). The goal of this work, as was introduced in MAPWeb validation, is to show a general agent framework that can be successfully used in several domains. So, in this case, we have considered evaluating the qualitative factors that define the control module of the RoboSkeleton agents, given that our main interest is, in this case, to verify the capability of the architecture to execute coordinate global behaviours, and not to develop high quality low level ones.

Section 4.1.2 introduced the fact that the main goal of RECCA is that the soccer players are able to keep a play strategy, for instance, 4-4-2, or any other one. This is a global parameter that must be defined. However, other three local elements were defined for each player: the role, the local leader, and the size of the play area associated to the player. From these four elements, we have performed several experiments to define the global one, i.e. the play strategy, and one of the local ones, the play area of each player. The goal of this experimentation is to define which is the more accurate one. This accuracy is obtained visually by the designer, by observing if the formation of the team is kept, and if each player is able to follow his local leader, or if the local leader is lost, and hence, formation is also lost.

#### 4.4.1 Defining the Play Strategy

For defining the play strategy, the structure 4-4-2 (four defenders, four midfield players, and two attackers) is used as a basic play strategy. This basic team is faced against other teams with different play strategy. In the first evaluation, the basic team plays against another with a 5-5 strategy, i.e. only two lines of players. When these two teams are playing, it was observed that while the basic team was able to keep the play strategy, the other team was not. The main problem appears when the players try to follow their local leader when both are located on the same play line (given that there are only two lines). So, the player places himself in his line horizontally with respect to his leader, while attacking or defending are actions that typically must be executed vertically in the field.

So the opposite case was tested, executing several matches of the basic team against another with a 4-3-2-1 play strategy. Even though this team maintains the team structure better than the 5-5 one, the increment in the number of lines increases the effort to maintain the team strategy. The 4-4-2, as in the previous case, is able to keep the formation of the team for a longer time. Therefore, we can conclude that the RECCA control architecture does it better when a 4-4-2 play strategy is followed, even though it can be instantiated with different ones.

#### 4.4.2 Defining the Play Area Size of the Players

For tuning the play area, the basic team tested in the previous test, with the 4-4-2 play strategy, is used. It is tested playing with different teams with the same play strategy but with different sizes of the play area of the different players:

- In the first experiment, the original team plays against another, where the play area of each player has been homogeneously increased. This new team seems to maintain the cohesion of the team better, given that the player can stay closer to each local leader, independent of the position of the global leader.
- In the second experiment, instead of increasing the play area of all the players homogeneously, the ones of the attackers and the defenders have been increased in a higher proportion than the midfield player ones. At the same time, the midfield areas have been modified to more accurate areas that try to minimize the effort of the team to keep the play strategy.

We can conclude that increasing the team area improves the capability of each player to follow its local leader, given that they can move in a higher area of the field. However, it can make the players keep very close among themselves, localizing all of them in a small area, which can not be good when playing real matches. This element shows the difficulty of defining a metric that allows us to empirically measure the goodness of a strategy, or certain parameters such as the area size of the players, and this is the reason we have not used it in this work. Testing them with real teams would require a finer tuning of low level skills that have not been carried out up to now.

## 5 Conclusions

This paper has presented two instantiations of SkeletonAgent, a flexible architecture for building MAS. The first instantiation, called MAPWEB, combines AI planning and Web information gathering techniques. We have used it to solve travel assistance problems in the e-tourism domain. The previous domain needs deliberative agents able to reason with the information gathered from the Web. The second instantiation, named ROBOSKELETON, implements a team of agents that can be executed in the RoboSoccer domain, following a reactive behaviour. With the adaptation of the architecture to these heterogeneous domains we show the flexibility of our approach, and how it can be adapted following the requirements of very different application domains. The flexibility of SkeletonAgent is achieved by the integration of classical AI problem solving techniques (like Planning and learning) in these domains.

From the experimental evaluation of MAPWEB and ROBOSKELETON showed in Subsections 3.4 and 4.4 the following conclusions can be addressed:

- The experiments carried out in Section 3.4 provide two main conclusions. On the one hand, it is possible to integrate different AI techniques into deliberative agents allowing us to reason with and manage the Web information obtaining a good performance (time response vs number of solutions) in the whole system. On the other hand, using these techniques it is possible to integrate the information provided from different sources. This experiment shows how the information stored in several data sources allows to find new solutions. Other experiments have been carried out by the authors using heterogeneous sources (i.e. using flights and train companies) obtaining similar experimental conclusions [Camacho *et al.*2001a].
- The RoboCup domain is an interesting domain where the SkeletonAgent architecture can be successfully used to design the different agents of the multi-agent system. It has been shown that different agents with different complexity can be implemented, and at least, follow the criterias defined by SkeletonAgent. For instance, in relation to the Multi-Agent architecture, the Robosoccer domain has shown it also requires a manager-coach-execution agent scheme, that is reflected by the Soccer server, the coach, and the players respectively. Furthermore, the agent architecture is useful too, even though sometimes not all the modules are required.

To test the generality of SkeletonAgent, it is being used to implement other MAS that wich integrate different AI techniques, such as Genetic Programming, over different domains, such as the lawn mover domain or distributed searching of electronic news. Figure 8 shows the different lines of work that are currently being studied using SkeletonAgent.

The characteristics of the new implemented systems can be summarized in the following:

- SimpleNews [Camacho and Aler] is a multiagent system that is able to look for news from several electronic-newspapers. This MAS has been

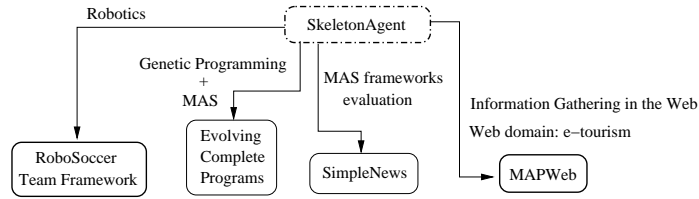


Figure 8: Several Multi-Agent Systems that have been implemented using the SkeletonAgent architecture.

implemented to test different performance characteristics such as time response, or scalability, with other well known agent architectures and frameworks like JATLite [Petrie1996], Jade [Bellifemine *et al.*1999], and ZEUS [Azarami and Thompson2000] in a common Web domain. The last two MAS frameworks are FIPA-compliant (<http://www.fipa.org>). Several versions of SimpleNews have been implemented using SkeletonAgent and previous MAS technologies.

- Genetic Programming and SkeletonAgent [Aler *et al.*2003]. A multiagent system has been implemented whose purpose is to build computer programs. Each agent in the multiagent system will be in charge of evolving a part of the program, which in this case can be either the main body of the program, or one of its different subroutines.

In the future several topics will be addressed with SkeletonAgent. At present, we are modifying systems like MAPWEB or ROBOSKELETON by adding new AI techniques. These new techniques can be used as new skills that could allow us to solve more complex problems by using information from the Web in the former system, or improving the soccer agents in the latter.

## References

- [Abowd *et al.*1997] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *ACM Wireless*, (3):421–433, 1997.
- [Aler *et al.*2003] Ricardo Aler, David Camacho, and Alfredo Moscardini. *Cooperation Between Agents to Evolve Complete Programs*, chapter In Intelligent Agent Software Engineering, pages 213–228. Valentina Plekhanova. University of Sunderland, United Kingdom. Ed. by Idea Group Publishing, 2003.
- [Ambite *et al.*2002] José Luis Ambite, Greg Barish, Craig A. Knoblock, Maria Muslea, Jean Oh, and Steven Minton. Getting from here to there: Interactive planning and agent execution for optimizing travel. In *The Fourteenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, 2002.

- [Azarami and Thompson2000] N. Azarami and S. Thompson. Zeus: A toolkit for building multi-agent systems. In *Proceedings of Fifth annual Embracing Complexity Conference*, April 2000.
- [Bellifemine et al.1999] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade - a fipa-compliant agent framework. In *Proceedings of the Conference on Practical Applications of Agents and Multi-Agents (PAAM'99)*, pages 97–108, April 1999.
- [Brenner et al.1998] W. Brenner, R. Zarnekow, and H. Wittig. *Intelligent Software Agents. Foundations and Applications*. Springer-Verlag. ISBN: 3-540-63411-8, New York, 1998.
- [Brooks1991] Rodney A. Brooks. Intelligence without representation. Number 47 in *Artificial Intelligence*, pages 139–159. 1991.
- [Camacho and Aler] David Camacho and Ricardo Aler. Software and performance measures for evaluating multi-agent frameworks. *Applied Artificial Intelligence*. Ed. by Taylor & Francis, In Press.
- [Camacho et al.2001a] David Camacho, Daniel Borrajo, José Manuel Molina, and Ricardo Aler. Flexible integration of planning and information gathering. In *Proceedings of the European Conference on Planning (ECP-01)*, Toledo, Spain, September 2001. Springer-Verlag. Series LNAI.
- [Camacho et al.2001b] David Camacho, César Hernández, and José M. Molina. Information classification in web agents using fuzzy knowledge for distance evaluation. In *WSES International Conference on: Fuzzy Sets & Fuzzy Systems (FSFS-01)*, Puerto De La Cruz (Tenerife), Spain, February 2001. IEEE.
- [Camacho et al.2005] David Camacho, Ricardo Aler, Daniel Borrajo, and José M. Molina. A multi-agent architecture for intelligent gathering systems. *AI Communications. The European Journal on Artificial Intelligence*. Ed. by IOS Press., To appear in 18(1), 2005.
- [Cheverst et al.2000] Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, and Christos Efstratiou. Developing a context-aware electronic tourist guide: some issues and experiences. In *CHI*, pages 17–24, 2000.
- [Dautenhahn1998] Kerstin Dautenhahn. The art of designing socially intelligent agents: science, fiction and the human in the loop. *Applied Artificial Intelligence Journal*, 1(7), 1998.
- [Decker and Sycara1997] Keith Decker and Katya Sycara. Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9:230–260, 1997.
- [Fernández and Borrajo2000] Fernando Fernández and Daniel Borrajo. VQQL. Applying vector quantization to reinforcement learning. In *RoboCup-99: Robot Soccer World Cup III*, number 1856 in *Lecture Notes in Artificial Intelligence*, pages 292–303. Springer Verlag, 2000.

- [Fernández *et al.*2000] Fernando Fernández, Germán Gutiérrez, and Jose Manuel Molina. Cooperación en sistemas distribuidos de robots reactivos minimizando la cantidad de información comunicada. In *Simposio Español de Informática Distribuida*, pages 61–68, 2000.
- [Fodor *et al.*2002] Oliver Fodor, Mirella Dell’Erba, Francesco Ricci, and Hannes Werthner. Harmonise: a solution for data interoperability. In *Proceedings of the 2nd IFIP Conference on E-Commerce, E-Business & E-Government*, October 7-9 2002.
- [Jennings *et al.*1998] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):275–306, 1998.
- [Kitano *et al.*1997] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The Robocup synthetic agent challenge. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI97)*, pages 24–49, San Francisco, CA, 1997.
- [Lesser1999] Victor R. Lesser. Cooperative multiagent systems: A personal view of the state of the art. *Knowledge and Data Engineering*, 11(1):133–142, 1999.
- [Matellán and Borrajo2001] Vicente Matellán and Daniel Borrajo. ABC<sup>2</sup> an agenda based multi-agent model for robots control and cooperation. *Journal of Intelligent and Robotic Systems*, 32(1):93–114, October 2001.
- [Noda1999] Itsuki Noda. *Soccer Server Manual*, version 4.02 edition, January 1999.
- [Norman *et al.*1997] Timothy Norman, Nick Jennings, Peyman Faratin, and Abe Mamdani. Designing and implementing a multi-agent architecture for business process management. In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Proceedings of the ECAI’96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III*, volume 1193, pages 261–276. Springer-Verlag: Heidelberg, Germany, 12–13 1997.
- [Petrie1996] Charles Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, 11(6):24–29, December 1996.
- [Petrie2000] Charles Petrie. Agent-based software engineering. In *Proceedings of PAAM 2000. The Practical Application of Intelligent Agents and Multi-Agents*, 2000.
- [Ricci and Werthner2002] Francesco Ricci and Hannes Werthner. Case base querying for travel planning recommendation. *Information Technology and Tourism Journal*, 4(3/4):215–226, 2002.

- [Riley and Veloso2001] Patrick Riley and Manuela Veloso. Coaching a simulated soccer team by opponent model recognition. pages 155–156, 2001.
- [Riley *et al.*2002] Patrick Riley, Manuela Veloso, and Gal Kaminka. An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 215–224. Springer-Verlag, 2002.
- [Schmidt-Belz] Barbara Schmidt-Belz. Personalized and location-based mobile tourism services.
- [Sommaruga *et al.*1996] Lorenzo Sommaruga, Nikos M. Avouris, and Marc Van Liedekerke. *Foundations of Distributed Artificial Intelligence*, chapter The evolution of the CooperA platform, pages 365–400. John Wiley. London, 1996.
- [Staab and Werthner2002] Steffen Staab and Hannes Werthner. Intelligent systems for tourism. *IEEE Intelligent Systems*, pages 53–66, November/December 2002.
- [Stone2000] P. Stone. *Layered Learning in Multiagent Systems*. MIT Pres, 2000.
- [Werthner and Klein1999] Hannes Werthner and Stefan Klein. *Information Technology and Tourism - A Challenging Relationship*. Springer-Verlag, 1999.
- [Zipf2002] Alexander Zipf. User-adaptive maps for location-based services (lbs) for tourism. In *Proceedings of the 9th International Conference for Information and Communication Technologies in Tourism, ENTER 2002*. Springer Computer Science. Heidelberg, Berlin, 2002.