Computing and Informatics, Vol. 36, 2017, 1019–1040, doi: 10.4149/cai_2017_5_1019

DYNAMIC MONITORING IN PANGEA PLATFORM USING EVENT-TRACING MECHANISMS

Luis Búrdalo, Andrés Terrasa, Vicente Julián

Department of Computer Systems and Computation
Universitat Politècnica de València, Spain
e-mail: {lburdalo, aterrasa, vinglada}@dsic.upv.es

Javier Bajo

Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid, Spain
e-mail: jbajo@fi.upm.es

Sara Rodríguez, Juan Manuel Corchado

Departamento de Informática y Automática
Universidad de Salamanca, Spain
e-mail: {srg, corchado}@usal.es

Abstract. The use of distributed multi-agent systems (MAS) have increased in recent years, with the growing potential to handle large volumes of data and coordinate the operations of many organizations. In these systems, each agent independently handles a set of specialized tasks and cooperates to achieve the goals of the system and a high degree of flexibility. Multi-agent systems have become the most effective and widely used form of developing this type of application in which communication among various devices must be both reliable and efficient. One of the problems related to distribute computing is message passing, which is related to the interaction and coordination among intelligent agents. Consequently, a multi-agent architecture must necessarily provide a robust communication platform and control mechanisms. This paper presents the integration of an event-tracing model in an agent platform called PANGEA. Adding this new capability, the platform allows improving the monitoring and analysis of the information that agents can send/receive in order to fulfil their goals more efficiently.

Keywords: Artificial intelligence, multi-agent systems, agent platforms, tracing systems, virtual organizations

Mathematics Subject Classification 2010: 68-T42

1 INTRODUCTION

Nowadays, the use and importance of multi-agent systems (MAS) has increased enormously due to their flexible behaviour, which is very useful to deal with complex problems in dynamic and distributed environments. This is not only due to agents individual features (like autonomy, reactivity or reasoning power), but also to their capability to communicate, cooperate and coordinate among them in the MAS in order to fulfil their goals.

These interactions generate a lot of knowledge supporting this social behaviour, which is referred to by Marik et al. in [1] as social knowledge. Social knowledge is one of the most important features that make MAS appropriate to deal with complex problems in dynamic and distributed environments. The key to this is the capacity of agents to communicate and coordinate with other agents in the MAS in order to get their objectives. This capacity, though based on high-level social concepts such as social commitments, trust, norms or reputation, is usually incorporated to the MAS at user level, using interchange messages policies or blackboard systems. This can produce overhead, reducing the scalability of the MAS. In addition, it has to be taken into account that sometimes it is difficult to trust information from other agents, especially in open MAS. An alternative solution to provide social knowledge could be an event tracing system, integrated within the multi-agent platform, which could be used by agents in the system to perceive and monitor their environment (actions that other agents do) without having to actively notify each change to the rest of the agents that could be interested in what they do. Such a tracing system, integrated within the multi-agent platform and providing a trustworthy event set which were capable to reflect not only communication among agents, but also agents perceptions, etc., could be used as a way to provide and manage social knowledge to the MAS. Moreover, adding a tracing system into a MAS platform could be used as a knowledge provider must not be human-oriented, but entity-oriented, so that these tracing entities (in a transparent way) are able to receive events and process them or incorporate them to their reasoning process at run time in order to take advantage from that.

In this way, this paper presents the integration of an event-tracing model [2] in an agent platform called PANGEA [3, 4, 5]. This platform is specially suited for developing Virtual Organizations. The concepts of roles, organizations and norms are fully supported by the platform assuring flexibility and scalability. The integration of this event-tracing model simplifies the work of tracing the information reducing its scalability as a centralized mechanism. The proposed approach pretends the use of event-tracing mechanisms in multi-agent systems, as an indirect interaction and coordination mechanism to improve the amount and quality of the information that agents can perceive in order to fulfil their goals more efficiently. Moreover, the event tracing system can help reducing the amount of unnecessary information in the MAS platform. These characteristics of the event-tracing model improve the way in which entities of the multi-agent system perceive each other and their environment. This improvement also facilitates the way in which high-level social abstractions can be developed and incorporated into a multi-agent system.

The proposed approach cannot be considered just a publish/subscribe system, which allows subscribers to filter events attending to attributes (content-based filtering). The proposed model does not rely on a single, centralized broker, but on a distributed manager, which is in charge of coordinating all the event tracing processes. This avoids excessive centralization, which may lead to bottlenecks and poorly scalable systems.

The remainder of the paper is structured as follows: the next section introduces some existing works regarding platforms and tracing. Section 3 presents an overview of the main characteristics of the PANGEA platform while Section 4 explains the proposed event-tracing framework. Section 5 provides a performance evaluation followed by Section 6, which finally, includes the conclusions.

2 RELATED WORK

It is possible to find in the literature different platforms for creating multi-agent systems and, from the perspective of the organizational theory, it is possible to establish two different categories: those platforms that simply provide support for the creation and execution of agents based on interactions, and those that incorporate organizational aspects such as norms and roles. It is necessary to remark that most of the existing agents platforms were designed from the first perspective and do not allow virtual organizations (VOs). This is the case of platforms such as FIPA-OS [7, 8], April Agent Platform (AAP) [9] or Jason [10, 11, 12], whose main contribution is the incorporation of the belief-desire-intention (BDI) model [26] into the agents architecture [13]. Probably the most commonly used platform is JADE (Java Agent DEvelopment Framework) [14], sometimes including the Jadex add-on [15]. These are platforms used to create agents that incorporate a deliberative architecture and knowledge representation mechanism and reasoning model based on logics. The platform provides a communication layer and a services management module. However, the design does not take organizational aspects into account.

A design directed by organizational concepts requires taking into consideration the normative and organizational aspects that the platform itself should provide. MadKit [16] was one of the first platforms designed to consider basic organizational aspects. Another pioneering platform in terms of organizational aspect was Jack Teams [17], which introduced the concept of "Team-oriented programming" as an intuitive paradigm to encapsulate coordination activity. S-Moise+ is a more sophisticated organizational middleware that evolves the Moise+ model [18, 19]. S-Moise+ can be used in conjunction with Jason to achieve a more complete model [20]. Hence the emergence of J-Moise+ [21], which is very similar to S-Moise+ regarding the overall system concepts. AMELI is a platform designed to work with institutions aimed at regulating organizations [22]. Another proposal that we can find in the literature is the THOMAS framework [1], which is based on the idea that architectural services (the services offered as functionalities by the agents on THOMAS) are offered as web services. As a result, the final product is entirely independent of any internal agent platform and fully addressed for open multi-agent systems [23]. One of the most complete and recent platforms that have been found in the literature review is Janus [24]. Janus is the evolution towards organizations of the platform previously known as TinyMAS (no longer under development).

In summary, to deal with all aspects of complex organizational systems, it is necessary to deal with concepts related to organization and openness, and to dynamically analyse the information flows within the platform, which is not the case for most of the existing solutions [25]. Besides, most of the existing platforms do not follow the same design standards, which makes difficult to provide compatibility with the information exchange. Each framework uses a different model file syntax and provides different libraries. In this paper, we focus on the PANGEA platform, designed to use standards that have already demonstrated their robustness and can be easily incorporated in existing platforms.

According to the analysis above, it is necessary to provide the PANGEA platform with an event-tracing mechanism to analyse the organizational dependencies and communicative acts between the social entities of the organization. Although PANGEA is an open system that can work with heterogeneous entities (different environments, languages or platforms), according to the analyzed related work, it is possible to observe that it is very important to analyze the flow of information among the base entities that make up the platform. We could observe that it was necessary to dynamically analyze information flows within the platform (between entities, organization and agents), which is not the case with most existing solutions, because it is at that point where it is possible to understand where to improve the platform and analyze possible bottlenecks or faults. It is necessary to provide the PANGEA platform with an event tracking mechanism to analyze these organizational dependencies and observe the workflow of its entities. If we analyse previous works in this area we can find one of the most popular tracing facilities for MAS, which is the Sniffer Agent provided by JADE [32]. This is a tool focused on establishing a trace of the communicative acts between the agents of the platform. The messages can be stored in a log file and analysed after the execution of the multi-agent system, that is, an execution of several agents within the platform, in a concrete case study, along with the base entities that form it. JADE also provides an Introspector Agent, which can be used to examine the life cycle of any agent in the system, its behaviours and the messages it has sent or received. Jadex [33] provides a Conversation Centre, which allows a user to send messages directly to any agent while it is executing and to receive answers to those messages from a user-friendly interface. One of the main contributions of Jadex is a BDI Tracer, which can be used to visualize the event of an agent, that is to observe the behavior of the agent (messages sending and receive, operations, and so on), while it is executing. The JACK [34] supports monitoring communication among agents by means of Agent Interaction Diagrams. It also provides other introspecting tools with different functionalities: a Design Tracing Tool, to view internal details of JACK applications during execution, and a Plan Tracing Tool, to display agent behaviour by tracing executing tasks in a graphical environment. Tasks consist of events and the plan instances that handle them. Each event that is traced within a task can be shown graphically, with nodes that are highlighted according to actions within the task. JACK also provides debugging tools that work at a lower level of abstraction in order to debug the multi-agent system in a more exhaustive way: Audit Logging, Generic Debugging/Agent Debugging. We can find other examples of tracing mechanism in the ZEUS platform [35] and Jason [12] to examine the internal state of agents and the messages exchange. Some of the existing platforms use tools provided by third party developers as the Java Sniffer [35], the ACLAnalyser [36, 37], or in the Prometheus methodology [39], which intercept, store and analyse messages exchanged by agents. Lam et al. [40] designed an iterative method based on tracing the event of an agent to visualize multi-agent applications to have a better understanding of the internal functioning of MAS. They also designed a Tracer Tool to support the proposed tracing method [41]. Bose et al. focus on a combination of a predicate logical Temporal Trace Language (TTL) for the formal specification and analysis of dynamic properties and a Checking Tool presented in [42] that enables the formal verification of properties against a set of traces.

As can be observed in the literature, tracing systems are useful tools in MAS, but have a limited use, as they have been conceived as debugging tools aimed at improving validation and verification processes. They have also been conceived as tracing tools that will help the user to understand the internal structure of the system. This way, the information is presented to the user to analyse the communication paths and improve the overall design of the system. Some multi-agent platforms provide their own tracing facilities, although there is also important work carried out by third party developers. The conclusions obtained in this section are that there is not a standard, general tracing mechanism, which lets agents, and other entities in the system trace each other as they execute, and there is a lack of this kind of mechanisms in organization-based multi-agent platform. Given the growing importance of organizational multi-agent systems, it is necessary to investigate new methods and tools to monitor and analyse the organizational aspects of MAS. In the next section we describe the main characteristics of the PANGEA platform that will be used as the basis to incorporate the new tracing mechanism.

3 THE PANGEA PLATFORM

In this section, we present PANGEA, a multi-agent platform especially designed to create, manage and control VOs, with advanced capacities for self-organization [3, 4, 5]. The platform has been designed to incorporate different agent architectures and propose a model of intelligent agent, CBR-BDI agent [26] to implement the coordination of re-organization strategies. The re-organization strategy is based on dynamic delegation and planning of tasks. The platform also incorporates tools for monitoring and controlling the life cycle of agents with graphic tools, and a communication protocol that allows broadcast communication, multicast according to the roles or suborganizations, or agent-to-agent. The platform was designed following a service-oriented perspective and facilitating the creation and management of organizational structures (with different topologies, suborganizations, etc). The combination of multiagent technology and Web Services has been a challenge since several years. In related works [44, 45, 46, 47] we can observe how the combination of a service oriented approach and multiagent platforms offers several enhancements regarding interoperability and availability. In PANGEA, norms were incorporated into the platform in terms of rules and regulatory mechanisms aimed at managing the creation, spreading and acceptance of social norms (e.g. maximum execution time in agents of a given organization, maximum number of agents in a suborganization or communication restrictions between suborganizations or the agents themselves). In addition, the platform provides a debugging tool, a module for interacting with FIPA-ACL agents, java programming and easily extensible with possibility of having agents in various platforms (Windows, Linux, MacOS, Android and IOS), and interface to oversee the organizations.

The platform is conceived as a virtual organization of agents, composed of suborganizations and a communication system that facilitates the interaction between the different components of the platform. The core agents of the platform facilitate the coordination of the virtual organization, as shown in Figure 1:

1. OrganizationManager: the agent responsible for the coordination of organizations and suborganizations. It is responsible for granting access to the agents into the virtual organization and for assigning and managing roles, permissions and responsibilities. To carry out these tasks, it works with the OrganizationAgent, which is a specialized version of this agent;
2. InformationAgent: it is responsible for the management of the information and knowledge of the organization and controls the information and the knowledge bases;
3. ServiceAgent: the agent responsible for recording and controlling the operation of services offered by the agents;
4. NormAgent: it is agent that ensures compliance with all the norms defined into the organization and decides the sanctions that can be applied when a norm is broken;

5. CommunicationAgent: this agent controls the communication layer of the platform and manages the interaction mechanisms between agents and organizations;
6. Sniffer: it manages the message history and filters information by controlling communication initiated by queries.

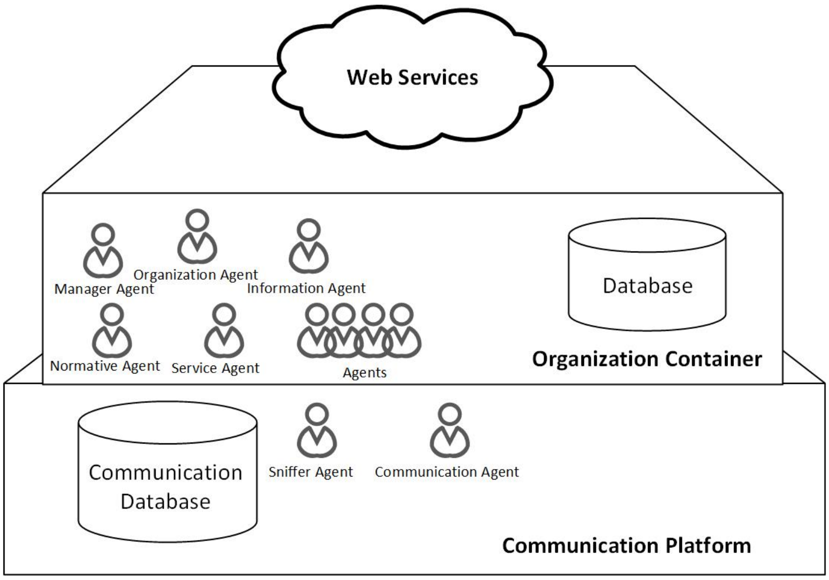

Figure 1. General view of the PANGEA architecture

As can be observed in Figure 1, PANGEA is a service-oriented platform where the agents distribute functionalities and work load between services. The capacities of the agents are designed as web services, so that they can coordinate different services. This makes it possible for the platform to include both a service provider agent and a consumer agent, thus emulating client-server architecture. The provider agent (a general agent that provides a service) knows how to contact the web service, the rest of the agents know how to contact with the provider agent due to their communication with the ServiceAgent, which contains this information about services.

One of the main features of this platform is the communication module and protocol. PANGEA will not pretend to present a new communication protocol; instead it will introduce the IRC protocol within multi-agent systems. The IRC protocol has been widely used in distributed environments demonstrating reliability and robustness. IRC protocol is used within the platform, providing its advantages such as ease of implementation and reliability, and its use confidence given that it has been widely used in online communities with good functionality. Moreover, the IRC standard allows the developed systems to handle a large number of connections and to ensure scalability [49]. Thanks to the proposed communication platform, the agents can be developed in any language that uses sockets to enable communication. The developed platform can create a general type of organization, and includes the possibility of creating open and highly dynamic systems, some examples are described in [3, 4, 5, 48, 49].

As shown in Figure 1, two main agents manage the communication system: the Communication Agent and the Sniffer. The former is responsible for checking connections to confirm that the agents are online and for interacting with the Norm-Agent to ensure that the agents comply with the norms in the organization. The latter is responsible for monitoring and recording all communicative acts. It also offers services so that other agents can obtain historical information, and facilitates the control of information flow for programmers and users. As before commented, one of the most important features that characterize the platform is the use of the IRC protocol (RFC1459 [27], RFC2810 [28], RFC2811 [29], RFC2812 [30] and RFC2813 [31]) for communication among agents. The open standard protocol enables its continuous evolution. There are also IRC clients for all operating systems, including mobile devices. The format of the messages includes the following blocks: prefix - command -parameters of the command \r \n. The prefix block may be optional in some messages, and required only for incoming messages. The command block is one of the originals from the IRC standard.

As it can be appreciated, the Sniffer Agent proposed in PANGEA is in charge of a very important task into the platform, but can be improved to provide a more precise and useful information about the organizational structure. Next section introduces how this work has been done without adding more overhead to the platform execution.

4 THE TRACING & SNIFFER FRAMEWORK

The Trace & Sniffer framework presented in this work has been designed to run on the PANGEA multi-agent platform. This framework provides a specific mechanism to obtain the information necessary for managing the message history and filtering information by controlling communication among entities. Specific mechanisms have been incorporated in the Sniffer Agent which, as above commented, is in charge of monitoring and recording all communicative acts. These tasks can be very hard in some environments with a high degree of social knowledge. For this reason, the proposed framework is an alternative and improved way for monitoring and analysing all the data flow among the agents of the system and it facilitates the decision making of corrective actions. The framework will provide mechanisms to analyse the behaviour of the implemented system from the point of view of the knowledge interchange. In order to describe how the proposed framework works, first we describe the main features of the event-tracing model, and the integration mechanisms designed to be used by agents of PANGEA and specifically by the Sniffer Agent.

4.1 The Event-Tracing Model

The proposed event-tracing model conceives the multi-agent system as a set of trace entities that share information by means of generating and receiving trace events. A trace entity is any component in the multi-agent system that is able to generate and receive trace events: agents, non-agents (artifacts, according to the definition in [43]), or aggregations of agent and non-agent entities. However, this work will only consider individual agents. A trace event is a piece of data that represents a significant computation that takes place during the execution of any component inside the multi-agent system. This model defines the following common attributes for each event:

- Event type: Trace events can be classified according to the nature of the information that they represent. So that, the rest of the data attached to the trace event can be interpreted.
- Time stamp: Global time at which the event took place; it is necessary to be able to chronologically sort events produced anywhere in the multi-agent system.
- Origin entity: The trace entity that originated the event.
- Attached data: Additional data that could be necessary to interpret correctly the trace event. The amount and type of these data will depend on the event type. Some trace events may not need any additional information.

Trace entities in the multi-agent system may participate in the tracing process by playing two different tracing roles: the Event Source (ES) role and the Event Receiver (ER) role. ES entities are those that generate trace events as they execute, while ER entities are those that receive these events. The relation between ES and ER entities is many-to-many: it is possible for events generated by an ES entity to be received by many ER entities; it is also possible for an ER entity to receive events from multiple ES entities simultaneously. These two tracing roles are not exclusive and any trace entity can play one or both of them at the same time.

The model defines an event-tracing protocol by which:

1. any agent can publish the types of events that it is able to generate (before generating them); and
2. any agent can subscribe to those trace events in which it is interested (before starting to receive them).

This protocol helps reduce as much as possible the overhead that tracing information can cause to the multi-agent system. ER entities must subscribe to those trace event types that they are interested in. Similarly, once an ER entity is not interested in receiving events of a type to which it had previously subscribed, the ER entity may unsubscribe from them. Consequently, only trace events of those types to which at least one ER has previously subscribed are generated and ER entities do not receive any tracing information in which they are not interested. This publication/subscription mechanism is dynamic in the sense that, at any time during the execution, agents can change their publications and subscriptions. In order to provide support to this publication/subscription mechanism, trace events are offered to agents in the system as trace services in a way similar to the way that traditional services are offered in the multi-agent system.

The model also considers a third tracing role, the Trace Manager role (TM). This role is responsible for controlling and coordinating the entire tracing process: registering tracing entities and event types, and giving support to the tracing and security models. This means that there must be at least one trace entity playing this role in order to give support to all these necessary features. The model establishes that the TM role can be played by a single entity or by a set of different entities in the multi-agent platform at the same time (in coordination) even in different nodes of the multi-agent system. Specifically, Trace Manager functions can be divided into four main tasks: to monitor all the tracing entities; to monitor and manage all of the active tracing services; to monitor and manage all the subscriptions to each tracing service; and, to control the authorization graph.

Regarding authorizations, when an ES entity publishes its trace events, it has also to specify which roles and/or entities in the multiagent system are authorized to receive such events. In this way, ES entities decide which ER entities can receive their trace events. This is defined as direct authorization. When an ER entity wants to receive events of a specific event type which come from a specific ES, it has to be authorized as an entity or it has to be able to assume one of the authorized roles. ER entities which are authorized to receive trace events from certain ES entity can also authorize other roles or entities to receive the same trace events. This is defined as authorization by delegation. In this way, the TM maintains an authorization graph for each event type which is being offered by each ES. This authorization graph is dynamic, since tracing entities can add and remove authorizations at run time. When an authorization, direct or by delegation is removed, all those delegated authorizations, which depended on the removed one, are also removed. The tracing system does not control the roles that entities can assume to receive trace events or to add and remove authorizations. In this sense, the MAS platform (PANGEA) will provide the necessary security mechanisms to prevent agents from assuming unappropriated roles.

A more detailed view of the TRAMMAS abstract model and the architecture model can be found in [2].

4.2 Integrating Event-Tracing in the PANGEA Platform

The tracing facilities described above have been incorporated to PANGEA by adding the needed infrastructure over a specific agent that plays the TM role. According to this design, agents have to send an ACL message to the TM agent whenever they want to publish or unpublish their available trace services and also when they want to subscribe to a trace service or to unsubscribe from it. The TM agent interacts with the PANGEA communication layer so that trace events generated by an agent are only injected into the network if there is an agent interested in receiving them; no trace event is received by an agent unless the agent has previously requested it.

The proposed integration follows a selective event tracing design where agents must subscribe to those trace events types which they are interested in. In the same way, once an agent is not interested in receiving events of a type to which it had previously subscribed, the agent may unsubscribe from them. The reason an agent decides to subscribe/unsubscribe to events relies on the internal decision making of the agent, which is domain-dependent.

In line with this design, the inclusion of the event-tracing model into a MAS system developed over PANGEA is relatively easy. As previously commented, a tracing service is a special service, which is offered by an ES entity to share its trace events. Therefore, the unique existing condition is that, as far as possible, an ES entity should implement its tracing service as a Web Service. This allows the ServiceAgent of PANGEA to offer the services to all the agents in the rest of suborganizations, and specifically for the Sniffer Agent. An EventTracing Suborganization has been included to create the tracing system. Figure 2 shows the agents and their relationships. This suborganization carry out the tasks that the TRAMMAS model assigns to the Trace Manager. Four agents form the suborganization, which cover the main functionalities of the Trace Manager:

- TraceEntityAgent in charge of registering and managing all the tracing entities.
- TracingServicesAgent in charge of registering and managing tracing services offered by ES entities.
- SubscriptionAgent, which stores and manages subscriptions to each tracing service and ES entity.
- AuthorizationAgent, which stores and manages the authorization needed for each tracing service and ES entity.

According to the proposed extension, PANGEA provides a communication layer for event-tracing, which allows agents to generate and receive trace events at run time. As a result, agents and other entities running on the PANGEA platform cannot only communicate in a direct way by means of ACL messages, but they can also communicate in an indirect way by means of trace events.

Moreover, PANGEA provides support to virtual organizations allowing for the development of open and dynamic multiagent systems, where agents are able to dynamically enter and leave the system, change their services, and change their relationships or the roles that they play in the organizations. The incorporation of the event tracing facilities improves how organization managers obtain certain information that is related to the organization performance at run time. The organizational knowledge that manager agents have can be used to estimate current state of the organization and propose possible changes. In this way, organization manager agents can retrieve all the information that is needed at each moment in a transparent way for the rest of the agents. Moreover, agents into the organization may also require some runtime information regarding the organization. These agents can also

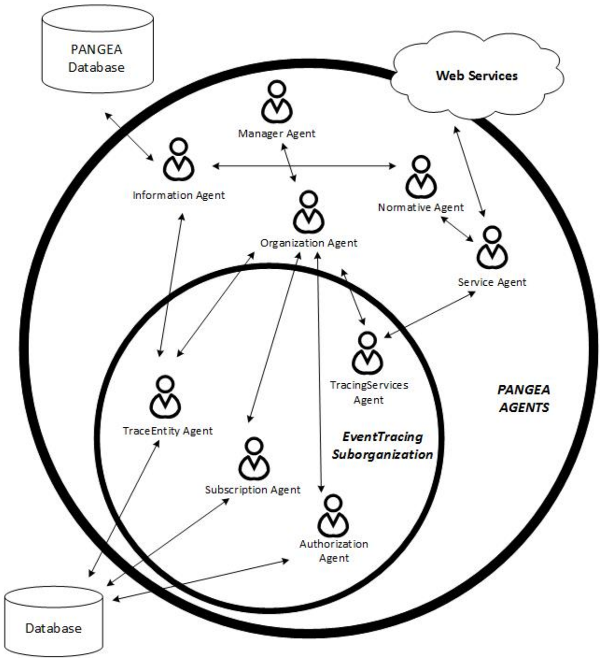

Figure 2. Visualization of the tracing suborganization architecture

subscribe to trace services and unsubscribe from them. In this way, organizational agents can carry out tasks that do not affect other agents in the organization and that do not require the supervision of the organization manager.

5 TRACING & SNIFFER PERFORMANCE EVALUATION

In order to test the Tracing & Sniffer framework, a case study was prepared to simulate a working environment. Four organizations were created to simulate four different departments within a company: accounting (composed of 4 accounting agents, one manager and 2 secretaries); quality control (composed of 2 evaluating agents and two training specialist agents); technical services (composed of 6 technical agents); and customer service (composed of 8 telephonist agents). According to the role of each agent, there are specific services offered that allow them to resolve the queries they receive.

In this proposed system, the client agent contacts the telephonist agent, which simply receives the requests and redirects it to the agent qualified to resolve the request. The telephonist agent extracts the key words from the message sent by the client and contacts the Services Agent to determine which agent can address the required service. At this point, from the point of view of the interactions among proposed agents, there exist different strategies that can be selected. Concretely, the telephonist agent can be implemented as an intermediary agent which can send the received queries to all the agents (acting as a Broadcast agent), or it can select, according to its own criteria, the appropriated agents to resolve the query and act as a Matchmaker agent. In this case, once the client is in contact with the appropriate agent, these agents can communicate without intervention of the telephonist agent. Another possibility is when the telephonist agent acts as a Broker agent. In this case the agent contacts only with the appropriated agents but also acts as an intermediary with the client during all the interaction process. Depending on the selected strategy differences among the number of messages interchanged can appear.

In order to analyse the possible differences among the proposed strategies that we can use to implement the above-proposed example we have executed different simulations. Specifically, four 30-minute simulations for each strategy (Broadcast, Matchmaker, Broker) were performed with 100 different types of requests randomly provided. Studying the Evaluation and Sniffer Agents it was possible to observe how both the simulation and message flow unfolded. Focusing specifically on the Sniffer, it is possible to obtain summary charts and diagrams, and specific metrics. Once the query is made, the Sniffer consults the database, filters the data and returns a URL that displays the desired data.

According to this, the Sniffer Agent can obtain the number of each type of message that a specific agent has received. Each message includes a tag that identifies the type of message, which makes it possible to filter information. With this information, it is possible to obtain a diagram of messages according to organization instead of agents (see Figure 3 as an example). Using the message identifier, it is also possible to see which agents processed a given request; using the Evaluation agents we can determine the number of requests processed by each agent.

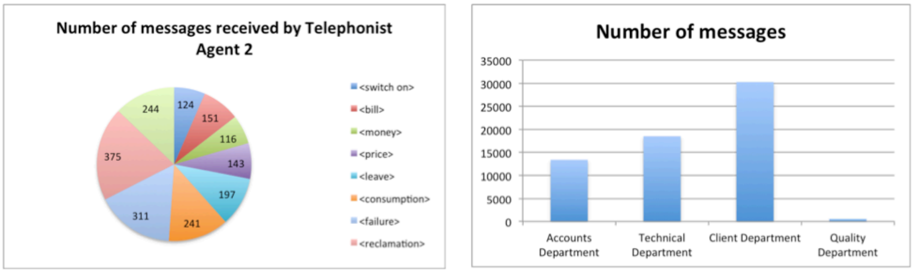

Figure 3. Examples of diagrams generated by the Sniffer Agent

The first experiment tried to measure the performance of the proposed system. In the proposed system, there exists an agent manager, which is responsible for managing in the company the organization dynamics. According to this, some experiments were executed regarding traffic reduction by using or not the tracing facilities. Concretely, the events managed by the manager agents were monitored. Figure 4 a) shows the number of events that were received by the manager in a static and a dynamically-steered monitoring strategy. In the static monitoring, we considered events represented as ACL messages. As can be observed, the number of events received is greater without using event tracing, and the differences between the two approaches becomes greater as the number of requests in the system increases. Therefore, dynamically-steered monitoring (in which only the required information is retrieved using event-tracing) considerably reduces the traffic load in the system. Note that in the static approach a lot of information is transferred that is finally not used by the manager. Finally, Figure 4 b) shows the average performance of the three approaches (Broadcast, Matchmaker, Broker) based on the messages exchanged in the whole organization. In this experiment, it can be observed that dynamically-steered monitoring clearly outperforms a static approach.

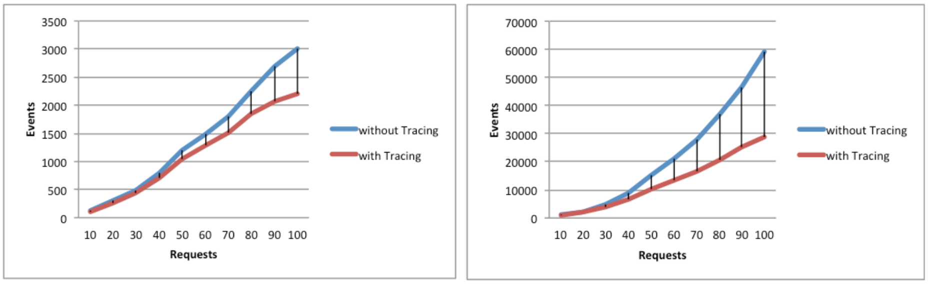

Figure 4. Number of events/messages received a) by the manager, or b) by the whole organization

The second experiment tried to show how the Sniffer Agent can also generate and analyse different metrics. This can be very useful to analyse which strategy is better than the rest in specific domains. So, in order to evaluate the different possible communication strategies, different metrics were considered. These metrics take into account the number of communication messages, their classification and the number of system messages. The set of metrics that automatically obtains the Sniffer Agent are the following:

- Effectiveness, which is calculated as the percentage of received OK messages respect all of the communication messages, delivered or not to those receiver agents which required them:

$$\text{Effectiveness} = \frac{\text{Messages}_{\text{OK}}}{\text{Messages}_{\text{OK}} + \text{Messages}_{\text{UNRCV}}} \tag{1}$$

  An effectiveness value of 100 % means that all required information was delivered to receivers which required it; while Effectiveness values under 100 % mean that some information, which was interesting for one or more receiver agents was never received by them and thus, these agents missed that information. Effectiveness does not provide any information regarding messages discarded by receiver agents; this is, those messages which were delivered to receiver agents and which were finally discarded as SPAM. For this reason, we add the following metric.

- Precision, which is calculated as the ratio between the number of relevant (OK) messages received and the total amount of received messages:

$$\text{Precision} = \frac{\text{Messages}_{\text{OK}}}{\text{Messages}_{\text{OK}} + \text{Messages}_{\text{SPAM}}} \tag{2}$$

  A Precision value of 100 % means that all of the information delivered to receiver agents was required by them and thus, receiver agents did not have to process and maybe discard any unrequired message, while Precision values under 100 % imply that some of the information delivered to receiver agents were unrequired. Precision does not show how these unrequired messages affect source agents, which have to send them anyway. It is necessary to reveal how the lack of precision of some strategies also affects source agents. In order to do so, we consider the following metric.

- Publishing Effort, which is calculated as the ratio between the number of OK messages received and the total amount of messages sent by all of the agents in the system:

$$\text{PublishingEffort} = \frac{\text{Messages}_{\text{OK}}}{\text{Messages}_{\text{SENT}}} \tag{3}$$

  A Publishing Effort value of 1 means that the number of communication messages sent by publisher agents and the number of required communication messages received by agents is the same: Agents do not waste any communication effort in order to deliver their information. Values between 0 and 1 reflect that agents transmit communication messages which are finally discarded once they arrive to their destination; while values over 1 reflect that the communication strategy helps publisher agents spreading their information, so that they do not have to send a copy of every message which is to be delivered.

- System Overhead, improvements in effectiveness and precision due to the use of certain strategies are not for free. Each strategy requires a different protocol to establish and maintain communication and thus, agents have to send and receive a different number of system messages depending on the strategy. We consider this traffic as system overhead, which can be viewed as the overhead incurred in the system to allow transferring data from the publisher onto the network and transferring this data from the network into the receiver. Instead of presenting the number of system messages injected in the system by any of the participants (publisher, receiver and middle agents), for each strategy a ratio between the number of system messages and the number of system messages using broadcast has been calculated:

$$\text{System Overhead} = \frac{\text{Messages}_{\text{SYS}}(\text{strategy})}{\text{Messages}_{\text{SYS}}(\text{Broadcast})} \tag{4}$$

where $\text{Messages}_{\text{SYS}}(\text{strategy})$ is the number of system messages using a specific strategy and $\text{Messages}_{\text{SYS}}(\text{Broadcast})$ is number of system messages using broadcast. Values between 0 and 1 would mean that the strategy required less system messages than using broadcast; however, none of the strategies does so.

	Effectiveness		Precision	
	Average	Deviation	Average	Deviation
Broadcast	100	0	0.48	0.54
Matchmaker	83.79	6.43	95.67	0.62
Broker	93.47	6.12	99.97	0.04

	Publishing Effort		System Overhead	
	Average	Deviation	Average	Deviation
Broadcast	0.004	0.005	1	0
Matchmaker	0.997	0.006	13.51	3.68
Broker	0.384	0.204	14.16	4.19

Table 1. Summary of Tests Results: four 30-minute simulations for each strategy (Broadcast, Matchmaker, Broker) with 100 different random requests

We can see how it has been performed each strategy analysing the obtained results generated by the Sniffer Agent for this concrete scenario (see Table 1). Obtained results have shown that Broadcast is an easy solution that obtains good results regarding effectiveness, but it has a very low precision and the publishing effort is very poor. On the opposite side, Matchmaker and Broker produce in general similar results. Nevertheless, the Broker approach offers better results in the effectiveness and precision metrics. In this situation, taking into account the results collected and generated by the Sniffer Agent, it seems that Broker is the best option to be used in this real scenario.

6 CONCLUSIONS

PANGEA is an architecture that has a great potential to create open systems, and more specifically, virtual agent organizations. This architecture includes different tools that make it easy for the end user to create, manage and control systems of this kind. One of the greatest advantages of PANGEA is the communication platform that, by using the IRC standard, offers a robust and widely tested system that can handle a large number of connections ensuring escalability, and that additionally facilitates the implementation for other potential extensions. One of the possible lacks of PANGEA is the use of the platform in the monitoring and on-line analysis of systems with a high demand in communication processes. In this sense, this paper has proposed an improved version of PANGEA adding a Tracing model, which simplifies the monitoring and storing processes of the Sniffer Agent. Thus, this agent can offer services that can be easily invoked to study and extract message information. The proposed improvement has been evaluated in a real scenario showing its potential to monitor and analyse implemented systems over the PANGEA platform.

Acknowledgements

This work is partially supported by the Ministerio de Economía y Competitividad of the Spanish Government and the European Regional Development Fund of the European Union (MINECO/FEDER TIN2015-65515-C4 & TIN2014-55206-R).

REFERENCES

[1] Marík, V.—Pechoucek, M.—Štepánková, O.: Social Knowledge in Multi-Agent Systems. Multi-Agent Systems and Applications (ACAI 2001). Lecture Notes in Computer Science, Vol. 2086, 2001, pp. 211–245, doi: 10.1007/3-540-47745-4_10.

[2] Búrdalo, L.—Terrasa, A.—Julián, V.—García-Fornes, A.: TRAMMAS: A Tracing Model for Multiagent Systems. Engineering Applications of Artificial Intelligence, Vol. 24, 2011, No. 7, pp. 1110–1119, doi: 10.1016/j.engappai.2011.06.010.

[3] Zato, C.—Villarrubia, G.—Sánchez, A.—Barri, I.—Rubión, E.—Fernández, A.—Rebate, C.—Cabo, J. A.—Alamos, T.—Sanz, J.—Seco, J.—Bajo, J.—Corchado, J. M.: PANGEA – Platform for Automatic CoNstruction of OrGanizations of IntElligent Agents. Knowledge and Information Systems, Vol. 29, 2011, No. 2, pp. 379–403.

[4] Villarrubia, G.—De Paz, J. F.—Bajo, J.—Corchado, J. M.: Ambient Agents: Embedded Agents for Remote Control and Monitoring Using the PANGEA Platform. Sensors, Special Issue Sensors Data Fusion for Healthcare, Vol. 14, 2014, No. 8, pp. 13955-13979, doi: 10.3390/s140813955, doi: 10.3390/s140813955.

[5] Zato, C.—Villarrubia, G.—Bajo, J.—Corchado, J. M.: An Integrated System for Disabled People Developed with the Agent Platform PANGEA. Advances in Distributed Computing and Artificial Intelligence Journal, Vol. 2, 2013, No. 3, pp. 65–77, doi: 10.14201/ADCAIJ2014266577.

[6] Argente, E.—Botti, V.—Carrascosa, C.—Giret, A.—Julián, V.—Rebollo, M.: An Abstract Architecture for Virtual Organizations: The THOMAS Approach. Knowledge and Information Systems, Vol. 29, 2011, No. 2, pp. 379–403, doi: 10.1007/s10115-010-0349-1.

[7] FIPA-OS Agent Platform. Emorphia, 2003.

[8] Poslad, S.—Buckle, P.—Hadingham, R.: The FIPA-OS Agent Platform: Open Source for Open Standards. Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, Vol. 355, 2000, p. 368.

[9] McCabe, F. G.—Clark, K. L.: April – Agent Process Interaction Language. International Workshop on Agent Theories, Architectures, and Languages (ATAL 1994). Lecture Notes in Computer Science, Vol. 890, 1994, pp. 324–340.

[10] Bordini, R. H.—Hübner, J. F.—Vieira, R.: Jason and the Golden Fleece of Agent-Oriented Programming. Multi-Agent Programming, Vol. 15, 2005, pp. 3–37, doi: 10.1007/0-387-26350-0_1.

[11] Bordini, R. H.—Hübner, J. F.: BDI Agent Programming in AgentSpeak Using Jason. Computational Logic in Multi-Agent Systems (CLIMA 2005). Springer Berlin Heidelberg, Lecture Notes in Computer Science, Vol. 3900, 2005, pp. 143–164, doi: 10.1007/11750734_9.

[12] Bordini, R. H.—Hübner, J. F.—Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Wiley Series in Agent Technology, Vol. 8, 2007.

[13] Rao, A. S.—Georgeff, M. P.: Modeling Rational Agents within a BDI-Architecture. Second International Conference on Principles of Knowledge Representation and Reasoning (KR '91), 1991, pp. 473–484.

[14] Bellifemine, F.—Poggi, A.—Rimassa, G.: JADE – A FIPA-Compliant Agent Framework. Proceedings of PAAM, London, Vol. 99, 1999, pp. 97–108.

[15] Pokahr, A.—Braubach, L.—Lamersdorf, W.: Jadex: A BDI Reasoning Engine. Multi-Agent Programming, Vol. 15, 2005, pp. 149–174, doi: 10.1007/0-387-26350-0_6.

[16] Gutknecht, O.—Ferber, J.: The MadKit Agent Platform Architecture. Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents (AGENTS 2000). Springer, Lecture Notes in Computer Science, Vol. 1887, 2000, pp. 48–55.

[17] JACK Intelligent Agents Teams Manual. Agent Oriented Software, 2004.

[18] Hübner, J. F.—Sichman, J. S.—Boissier, O.: A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. Brazilian Symposium on Artificial Intelligence (SBIA 2002). Lecture Notes in Computer Science, Vol. 2507, 2002, pp. 118–128, doi: 10.1007/3-540-36127-8_12.

[19] Hübner, J. F.—Sichman, J. S.—Boissier, O.: S-Moise+: A Middleware for Developing Organised Multi-Agent Systems. COIN I, Springer, Lecture Notes in Artificial Intelligence, Vol. 3913, 2006, pp. 64–78.

[20] Hübner, J. F.—Bordini, R. H.—Picard, G.: Using Jason and Moise+ to Develop a Team of Cowboys. International Workshop on Programming Multi-Agent Systems Lecture Notes in Artificial Intelligence (ProMAS 2009). Lecture Notes in Computer Science, Vol. 5442, 2008, pp. 238–242.

[21] Hübner, J. F.: Programming Organisational Agents with Moise+ & Jason. Technical Fora Group at EUMAS, Vol. 7, 2007.

[22] Sierra, C.—Rodriguez-Aguilar, J. A.—Noriega, P.—Esteva, M.—Arcos, J. L.: Engineering Multi-Agent Systems as Electronic Institutions. European Journal for the Informatics Professional, Vol. 4, 2004, No. 4, pp. 33–39.

[23] Giret, A.—Julián, V.—Rebollo, M.—Argente, E.—Carrascosa, C.—Botti, V.: An Open Architecture for Service-Oriented Virtual Organizations. International Workshop on Programming Multi-Agent Systems (ProMAS 2009). Lecture Notes in Computer Science, Vol. 5919, 2009, pp. 118–132.

[24] Galland, S.—Gaud, N.—Rodriguez, S.—Hilaire, V.: Janus: Another Yet General-Purpose Multiagent Platform. Seventh Agent-Oriented Software Engineering Technical Forum (TFGAOSE-10), Agent Technical Fora, Paris, `http://www.pa.icar.cnr.it/cossentino/AOSETF10/docs/galand_ppt.pdf`, 2010.

[25] Cossentino, M.—Gaud, N.—Hilaire, V.—Galland, S.—Abderrafiâa, K.: ASPECS: An Agent-Oriented Software Process for Engineering Complex Systems. Autonomous Agents and Multi-Agent Systems, Vol. 20, 2010, No. 2, pp. 260–304, doi: 10.1007/s10458-009-9099-4.

[26] Corchado, J. M.—Glez-Bedia, M.—De Paz, Y.—Bajo, J.—De Paz, J. F.: Replanning Mechanism for Deliberative Agents in Dynamic Changing Environments. Computational Intelligence, Vol. 24, 2008, No. 2, pp. 77–107, doi: 10.1111/j.1467-8640.2008.00323.x.

[27] Oikarinen, J.—Reed, D.: Internet Relay Chat Protocol. 1993, doi: 10.17487/rfc1459.

[28] Kalt, Ch.: Internet Relay Chat: Client Protocol. 2000.

[29] Kalt, Ch.: Internet Relay Chat: Server Protocol. 2000.

[30] Kalt, Ch.: Internet Relay Chat: Channel Management. 2000.

[31] Kalt, Ch.: Internet Relay Chat: Architecture. Technical Report, 2000.

[32] Bellifemine, F.—Caire, G.—Trucco, T.—Rimassa, G.—Mungenast, R.: JADE Administrator's Guide. TILab (February 2006), 2003.

[33] Pokahr, A.—Braubach, L.: Jadex Tool Guide. 2008.

[34] Agent Oriented Software Pty Ltd.: JACK Intelligent Agents Tracing and Logging Manual. 2010.

[35] Collis, J. C.—Ndumu, D. T.—Nwana, H. S.—Lee, L. C.: The ZEUS Agent Building Tool-Kit. BT Technology Journal, Vol. 16, 1998, No. 3, pp. 60–68.

[36] Botía, J. A.—Hernansáez, J. M.—Skarmeta, F. G.: Towards an Approach for Debugging MAS Through the Analysis of ACL Messages. German Conference on Multiagent System Technologies (MATES 2004). Lecture Notes in Computer Science, Vol. 3187, 2004, pp. 301–312.

[37] Botía, J. A.—Hernansáez, J. M.—Gómez-Skarmeta, A. F.: On the Application of Clustering Techniques to Support Debugging Large-Scale Multi-Agent Systems. International Workshop on Programming Multi-Agent Systems (ProMAS 2006). Lecture Notes in Computer Science, Vol. 4411, 2006, pp. 217–227.

[38] Tichy, P.—Slechta, P.: Java Sniffer 2.7 User Manual. 2006.

[39] Padgham, L.—Winikoff, M.—Poutakidis, D.: Adding Debugging Support to the Prometheus Methodology. Engineering Applications of Artificial Intelligence, Vol. 18, 2005, No. 2, pp. 173–190, doi: 10.1016/j.engappai.2004.11.018.

[40] Lam, D. N.—Barber, K. S.: Comprehending Agent Software. Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2005, pp. 586–593, doi: 10.1145/1082473.1082562.

[41] Lam, D. N.—Barber, K. S.: Debugging Agent Behavior in an Implemented Agent System. International Workshop on Programming Multi-Agent Systems (ProMAS 2004). Lecture Notes in Computer Science, Vol. 3346, 2004, pp. 104–125.

[42] Bosse, T.—Jonker, C. M.—Van Der Meij, L.—Sharpanskykh, A.—Treur, J.: Specification and Verification of Dynamics in Cognitive Agent Models. IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT '06), 2006, pp. 247–254, doi: 10.1109/IAT.2006.112.

[43] Omicini, A.—Ricci, A.—Viroli, M.: Artifacts in the A & A Meta-Model for Multi-Agent Systems. Autonomous Agents and Multi-Agent Systems, Vol. 17, 2008, No. 3, pp. 432–456, doi: 10.1007/s10458-008-9053-x.

[44] Buhler, P.—Vidal, J. M.—Verhageni, H.: Adaptive Workflow = Web Services + Agents. Proceedings of the First International Conference on Web Services, Nevada, 2003, pp. 1–7.

[45] Buhler, P. A.—Vidal, J. M.: Towards the Synthesis of Web Services and Agent Behaviors. Proceedings of the Agentcities: Challenges in Open Agent Environments Workshop (Corresponding with AAMAS 2002), ACM, Bologna, 2002, pp. 25–29.

[46] Rodríguez, S.—Julián, V.—Bajo, J.—Carrascosa, C.—Botti, V. J.—Corchado, J. M.: Agent-Based Virtual Organization Architecture. Engineering Applications of Artificial Intelligence, Vol. 24, 2011, No. 5, pp. 895–910.

[47] Bajo, J.—Corchado, J. M.—de Paz, V.—de Paz, J. F.—Rodríguez, S.—Martín, Q.—Abraham, A.: SHOMAS: Intelligent Guidance and Suggestions in Shopping Centres. Applied Soft Computing, Vol. 9, 2009, No. 2, pp. 851–862.

[48] Sánchez, A.—Villarrubia, G.—Zato, C.—Rodríguez, S.—Chamoso, P.: A Gateway Protocol Based on FIPA-ACL for the New Agent Platform PANGEA. Trends in Practical Applications of Agents and Multiagent Systems. Springer, Advances in Intelligent Systems and Computing, Vol. 221, 2013, pp. 41–51, doi: 10.1007/978-3-319-00563-8_6.

[49] Zato Domínguez, D. C.: Model for WCET Prediction, Scheduling and Task Allocation for Emergent Agent-Behaviours in Real-Time Scenarios. Ph.D. Dissertation. `http://gredos.usal.es/xmlui/handle/10366/125975`, University of Salamanca, 2014.

Luis Búrdalo received his Ph.D. in computer science and artificial intelligence from the Politechnical University of Valencia in 2015. He studied computer science at the Universitat Politecnica de Valencia (UPV), Spain, where he received his B.Sc. degree in 2004. Currently, he works as Researcher at the Department of Information Systems and Computation (DSIC) of the UPV. Also, he is a member of the Group of Information Technology/Artificial Intelligence (GTI/IA) research group. His research interests mainly include real-time systems, real-time artificial intelligence, and multiagent systems.

Andrés Terrasa studied computer science at the Universitat Politecnica de Valencia (UPV), Spain, where he received his B.Sc. degree in 1995, and his Ph.D. degree in 2001. He currently works as Associate Professor at the Department of Information Systems and Computation (DSIC) of the UPV. Also, he is a member of the Group of Information Technology/Artificial Intelligence (GTI/IA) research group. His research interests mainly include real-time systems, realtime artificial intelligence, and multi-agent systems.

Vicente Julián holds a position of Associate Professor of computer science at the UPV where he has taught since 1996. He is a member of the GTI-IA research group, and Deputy Director of the Official Master in Artificial Intelligence, Pattern Recognition and Digital Imaging at the UPV. Four international projects, two international excellence networks, twenty one Spanish projects and four technology transfer projects have covered the research on Artificial Intelligence. He has more than 50 works published in journals with outstanding positions in the list of the Journal Citation Reports, or published in conference proceedings that have a system of external peer review and dissemination of knowledge comparable to journals indexed in relevant positions. Moreover, he has more than 130 contributions and an h-index of 20, with more than 1 400 citations to his published work. He has supervised eight Ph.D. thesis.

Javier Bajo received his Ph.D. in computer science and artificial intelligence from the University of Salamanca in 2007. At present he is Associate Professor at the Politechnical University of Madrid (Spain). He received an Information Technology degree at the University of Valladolid (Spain) in 2001 and an engineering in computer sciences degree at the Pontifical University of Salamanca in 2003. He has been member of the organising and scientific committee of several international symposiums such as CAEPIA, IDEAL, HAIS, etc. and co-author of more than 100 papers published in recognized journals, workshops and symposiums.

Sara Rodríguez is Assistant Professor of computer science at University of Salamanca. She received her Ph.D. in computer science from this university in 2010. She received a Technical Engineering in systems computer sciences degree in 2004, and Engineering in computer sciences degree in 2007. Moreover, she has pursued other postgraduate degrees such as Master in the development of systems for e-commerce and Master in computer graphics animation. She has participated as a co-author in papers published in recognized international journals and she has published over 70 peer-reviewed articles in a range of topics from agent and multiagent systems, application of hybrid systems on bioinformatics, computer graphics animation, artificial vision, and so on.

Juan Manuel Corchado received his Ph.D. in computer science from the University of Salamanca in 1998 and his Ph.D. in artificial intelligence from the University of the West of Scotland (UK) in 2000. At present he is Director of the Biomedicine, Intelligent Systems and Educational Technology Group and Director of the M.Sc. Programs in E-commerce and Digital Animation of the University of Salamanca (Spain), previously he was Sub-Director of the Escuela Superior de Ingenieria Informatica of the University of Vigo (Spain, 1999–2000) and Researcher at the University of Paisley (UK, 1995–1998). He has been a research collaborator with the Plymouth Marine Laboratory (UK) since 1993. He worked on several Artificial Intelligence (AI) Research projects sponsored by Spanish and European public and private institutions and supervised seven Ph.D. students. He is the co-author of over 200 books, book chapters, journal papers, technical reports, etc., most of them present practical and theoretical achievements of AI Systems.