An Integrated System for Disabled People Developed with the Agent Platform PANGEA

Carolina Zato[a], Gabriel Villarrubia[a], Javier Bajo[b], Juan M. Corchado[a]

[a] Departamento Informática y Automática, Universidad de Salamanca
[b] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid

KEYWORD

Open multi-agent system
Virtual organizations
Agent platform
IRC standard
Distributed systems
Web services

ABSTRACT

New trends in multi-agent systems call for self-adaptation and high dynamics, hence the new model of open MAS or virtual organization of agents. However, as existing agent platforms are not yet equipped to support this behavior, it is necessary to create new systems and mechanisms to facilitate the development of these new architectures. This article presents PANGEA, an agent platform to develop open multi-agent systems, specifically those including organizational aspects such as virtual agent organizations. The platform allows the integral management of organizations and offers tools to the end user. Additionally, it includes a communication protocol based on the IRC standard, which facilitates implementation and remains robust even with a large number of connections. The introduction of a CommunicationAgent and a Sniffer make it possible to offer Web Services for the distributed control of interaction. In order to test PANGEA, an integral system was developed to help the disabled, gathering a set of easily deployable and integrated services under a single architecture.

1 Introduction

Nowadays, one of the research lines that the multi-agent systems are following is directed toward ensuring that these systems become more open and dinamic. An open MAS [BAJO, J. *et al*. 2009] should allow for the interaction between heterogeneous agents, which change over time, and architectures and even different languages. Because of their inherent changing nature, we cannot rely on agents' behaviour when it is necessary to establish controls on the basis of norms or social rules. For this reason, and because of the characteristics of open environments, new approaches are needed to support evolutive systems and to facilitate their growth and runtime updates. The dynamics of open environments is one of the reasons that have encouraged the use of Virtual Organizations of Agents (VOs). A VO [FERBER, J. *et al*. 2008] [FOSTER, I. *et al*. 2001] is an open system designed for grouping; it allows for the collaboration of heterogeneous entities and provides a separation between the form and function that define their behaviour.
VOs are conceived as a set of agents with roles and rules that determine their behavior and where previously established capabilities play a crucial role. Possible topologies and organizational aspects well as their communication and coordination mechanisms determine largely the flexibility, dynamism and openness that the multi-agent system can offer. The concept of organization is seen as a promising solution to manage the coordination of the agents and control their behaviours and actions. Every organization needs coordination support to determine explicitly how to organize and carry out the actions and tasks within it.
There are different platforms, which will be shown later, that allow for the creation of multi-agent systems. These platforms greatly facilitate the task of working with agents. However, in terms of platforms that allow for the creation of a VO, the number is drastically reduced; it is in fact difficult to find a single platform that covers all the requirements that a VO requires,

such as reorganization and adaptation facilities, norm compliance, different organizational topologies, service and role management.

At the same time, distributed multi-agent systems have become increasingly sophisticated in recent years, with a growing potential to handle large volumes of data and coordinate the operations of many organizations [HELSINGER, A. *et al*. 2004]. Distributed intelligent systems are based on the use of cooperative agents, where each agent independently handles a small set of specialized tasks and cooperates to achieve the system-level goals and a high degree of flexibility [GRUVER, A. 2004]. Another problem for distributed systems is that they must now be able to accept requests from different devices. There is, furthermore, a rapidly growing use of mobile devices with limitless connection possibilities and computing power. From a practical perspective, multi-agent systems face yet another challenge: obtaining light intelligent agents that can be deployed in this type of device without relinquishing their capabilities.

For the aforementioned reasons, PANGEA introduces a new protocol based on the IRC standard to facilitate communication in distributed multi-agent systems. This protocol facilitates cooperation between agents, which is critical in these kinds of systems. It includes a robust communication model that allows intelligent agents to connect from a variety of devices. This communication is easy to implement and enables agents deployed in different devices to cooperate. These agents can be developed both quickly and in any language using the tools provided by the platform. PANGEA also includes facilities for implementing VOs and suborganizations, following any topology and with the appropiate tools for managing the VO itself. It is important to highlight that PANGEA is not a framework or a simple tool; it is a complete platform for the execution and management of VOs.

The remainder of the paper is structured as follows: the next section introduces some existing platforms. Section 3 presents an overview of the main characteristics of the platform and the communication protocol. Section 4 presents the case study and finally, the last section includes the conclusions.

2 Related Works

All platforms for creating multi-agent systems that exist to date should be studied according to two principal categories: those that simply support the creation and interaction of agents, and those that permit the creation of virtual organizations with such key concepts as norms and roles.

Initially, most of the agents' platforms do not allow VOs. This is the case of FIPA-OS [EMORPHIA, 2013] [POSLAD, S. *et al*. 2000], April Agent Platform (AAP) [MCCABE, A. 1995] and JASON [BORDINI, R. *et al*. 2005] [JASON, 2013] [BORDINI, R. *et al*. 2007], which its main contribution is the easy with which belief-desire-intention (BDI) [CORCHADO, J. *et al*. 2008] agents can be implemented [RAO, A. *et al*. 1991]. In practice, the platform that is most commonly used to develop multi-agent systems in real case studies is JADE (Java Agent DEvelopment Framework) [BELLIFEMINE, F. *et al*. 1999] and Jadex [POKAHR, F. *et al*. 2005]. Jadex is a software framework and a extension of JADE for the creation of goal-oriented agents following the BDI model.

One of the more recent platforms is JIAC [HIRSCH, B. *et al*. 2009] [LUTZENBERGER, M. *et al*. 2009], a service-aware framework. JIAC V is a Java based agent framework with an emphasis on industrial requirements such as software standards, security, management, and scalability. It combines agent technology with a service oriented approach. Together with the framework, a new language called JADL++ was created; this platform stands out because of its service matching capabilities and an excellent usability.

Until now, these platforms can create agents (some with different models), follow their life cycle and manage communication and services. However, in the case of VO, it is necessary to take into account the normative and organizational aspects that the platform itself should provide.

MadKit [GUTKNECHT, O. *et al*. 2000] was one of the first platforms to consider basic organizational aspects. The platform architecture is rooted in the AGR (agent-group-role) model [GUTKNECHT, O. *et al*. 1997] developed in the context of the AALAADIN project. Another pioneering platform in terms of its structural aspect was Jack Teams [JACK,

2005], which introduced the concept of "Team-oriented programming" as an intuitive paradigm to encapsulate coordination activity. The JACK Teams extension introduces the new constructs team, role, teamdata, and teamplan. Various authors have used JACK Teams in their research, notably [BISHT, A. *et al.* 2007] [JARVIS, J. *et al.* 2006]. The main disadvantage of this platform is that it only allows hierarchical team structures.
S-MOISE+ is an organizational middleware that follows the MOISE+ model [HUBNER, J. *et al.* 2002] [HUBNER, J. *et al.* 2006]. In MOISE+ a multi-agent system is specified as an organization, distinguishing three main aspects. The structural aspect is in charge of the structure or topology of the organization. The functional aspect is in charge of the organizational operations and the tasks that should be carried out. And the normative aspect specifies the permission and obligations of each role. The printed material used for this research includes systems that were developed in conjunction with JASON, using S-MOISE+ as the middleware to achieve a more complete model [HUBNER, J. *et al.* 2009]. Hence the emergence of J-Moise+ [HUBNER, J. 2006], which is very similar to S-Moise+ regarding the overall system concepts.
AMELI is a middleware that can work with electronic institutions (similar to VO) [SIERRA, C. *et al.* 2004]. An innovative feature of AMELI is its general purpose; because it can interpret any institution specification, organization or topology, it can be regarded as domain-independent, which is the reason why AMELI offers a higher (social) level of abstraction [ESTEVA, M. *et al.* 2003].
One of the main disadvantages of platforms geared toward VO is that the concept of service is somewhat diminished, which impacts the management associated with these services and the Directory Facilitator (DF) described in the FIPA standard. This need led to the creation of the THOMAS framework [BAJO, J. *et al.* 2010]. THOMAS is based on the idea that no internal agents exist and architectural services 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 [GIRET, A. 2009].
Until now, all tools used to create, manage and control VOs are frameworks or middlewares that require other agent platforms to be able to integrally develop a VO. This implies that the deficiencies of agent platforms are further diffused if the layer for managing the VO cannot solve the problem.
One of the most complete and recent platforms that have been found in the literature review is Janus [GALLAND, S. 2010]. Janus is the evolution towards organizations of the platform previously known as TinyMAS (no longer under development.). This platform was specially designed to deal with the implementation and deployment of holonic and multi-agent systems (HMAS) [GAUD, N. *et al.* 2008]. The key aspects handled by the platform are organizations, roles, interactions and capacities. However, it disregards the concept of norm and service. The platform does not explicitly consider the normative aspects of the organizations; they are instead included within the concept of role.
In summary, to deal with all aspects of complex systems, MAS such as VO, it is necessary to deal with multiple levels of abstractions and openness, which is not the case for most solutions [COSSENTINO, M. *et al.* 2010]. Moreover, although the agent frameworks or platforms have similarities, there are subtle diferences, too. Each framework uses a different model file syntax and provides different libraries. In our platform, we have tried to use standards that have already demonstrated their robusteness and are known within the research community, making the implementation of new architectures easier and highly reliable.

3 PANGEA Overview

As previously mentioned, a platform that can integrally create, manage and control VOs was developed for this study. In general terms, the proposed platform includes the following characteristics:

- Different models of agents, including a BDI and CBR-BDI architecture [CORCHADO, J. *et al.* 2008].
- Ability to control the life cycle of agents with graphic tools.
- A communication protocol that allows broadcast communication, multicast according to the roles or suborganizations, or agent to agent.
- A debugging tool.

- Module for interacting with FIPA-ACL agents.
- Service management and tools for discovering services.
- Web services.
- Flexibility in allowing organizations with any topology and suborganizations.
- Organization management.
- Services for dynamically reorganizing the organization [ZATO, C. *et al*. 2012].
- Services for distributing tasks and balancing the workload [ZATO, C. *et al*. 2012].
- A business rules engine to ensure compliance with the standards established for the proper operation of the organization.
- Java programming and easily extensible.
- Possibility of having agents in various platforms (Windows, Linux, MacOS, Android and IOS)
- Interface to oversee the organizations.

In PANGEA, the roles, norms and the organizations themselves are main classes that facilitate the inclusion of organizational aspects. The services are also included as classes completely separate from the agent, facilitating their flexibility and adaption. The Capacity service determines the reasoning mechanisms available to the agent.

When launching the main container of execution, the communication system is initiated; the agent platform then automatically provides the agents shown in Figure 1 to facilitate the control of the organization:

- OrganizationManager: the agent responsible for the actual management of organizations and suborganizations. It is responsible for verifying the entry and exit of agents, and for assigning roles. To carry out these tasks, it works with the OrganizationAgent, which is a specialized version of this agent.
- OrganizationAgent: it is a specialized version of the OrganizationManager, which is introduced automatically in each suborganization to help the OrganizationManager and avoid its overload.
- InformationAgent: the agent responsible for accessing the database containing all pertinent system information.
- ServiceAgent: the agent responsible for recording and controlling the operation of services offered by the agents. It works as the Directory Facilitator defined in the FIPA standar.
- NormAgent: the agent that ensures compliance with all the refined norms in the organization.
- CommunicationAgent: the agent responsible for controlling communication among agents, and for recording the interaction between agents and organizations.
- Sniffer: manages the message history and filters information by controlling communication initiated by queries.

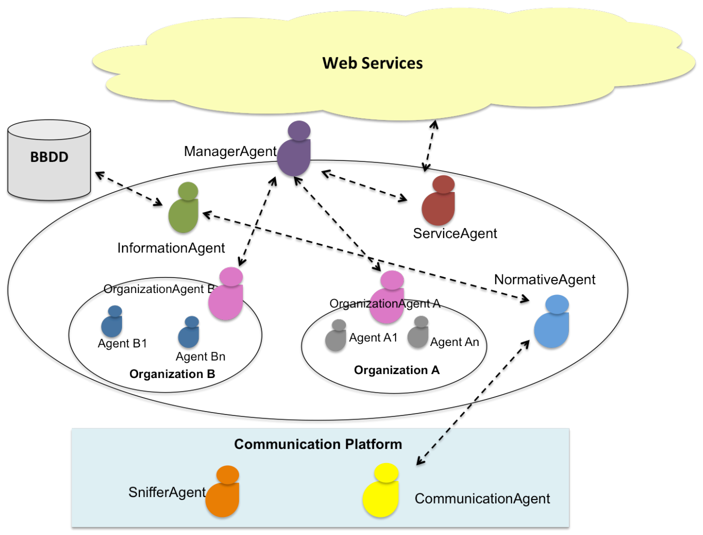


Fig. 1. PANGEA agents

The platform enables two modes of operation. In the first mode, the agents reside in the machine itself, while in the second mode the platform allows for the possibility of initiating all agents in different machines. The latter case has the disadvantage of allowing only minimal human intervention since it is necessary to previously specify the address of the machine where each of the agents are to reside; however it has the advantage of greater system distribution.

We have created a service-oriented platform that can take maximum advantage of the distribution of resources. To this end, all services are implemented as Web Services. This makes it possible for the platform to include both a service provider agent and a consumer agent, thus emulating a client-server architecture. The provider agent (a general agent that provide 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 informacion about services.

Once the client agent's request has been received, the provider agent extracts the required parameters and establishes contact. Once received, the results are sent to the client agent. Using Web Services also allows the platform to introduce the SOA architecture (Service-oriented Arquitecture) [JOSUTTIS, N. *et al*. 2007] into MAS systems. SOA is an architectural style for building applications that use services available in a network such as the web. It promotes loose coupling between software components so that they can be reused. Applications in SOA are built based on services. A service is an implementation of a well-defined functionality, and such services can then be consumed by clients in different applications or processes. SOA allows for the reuse of existing services and a level of flexibility that was not possible before in the sense that:

- Services are software components with well-defined interfaces that are implementation-independent. An important aspect of SOA is the separation of the service interface from its implementation. Such services are consumed by agent clients that are not concerned with how these services will execute their requests.
- Services are self-contained and loosely coupled encouraging independence.
- Services can be dynamically discovered
- Composite services can be built from aggregates of other services [WU, D. *et al*. 2003].

Each suborganization or work unit is automatically provided with an OrganizationAgent by the platform during the creation of the suborganization. This OrganizationAgent is similar to the OrganizationManager, but is only responsible for controlling the suborganizationn, and can communicate with the OrganizationManager if needed. If another suborganization is created hierarchically within the previous suborganization, it will include a separate OrganizationAgent that communicates with the OrganizationAgent from the parent organization. These agents are distributed hierarchically in order to free the OrganizationManager of tasks. This allows each OrganizationAgent to be responsible for a suborganization although, to a certain extent, the OrganizationManager can always access information from all of the organizations. Each agent belongs to one suborganization and can only communicate with the OrganizationAgent from its own organization; this makes it possible to include large suborganizational structures without overloading the AgentManager. All of the OrganizationAgents from the same level can communicate with each other, unless a specific standard is created to prevent this. One possible topology is shown in Figure 2, with the ManagerAgent establishing communication with the OrganizationManager.

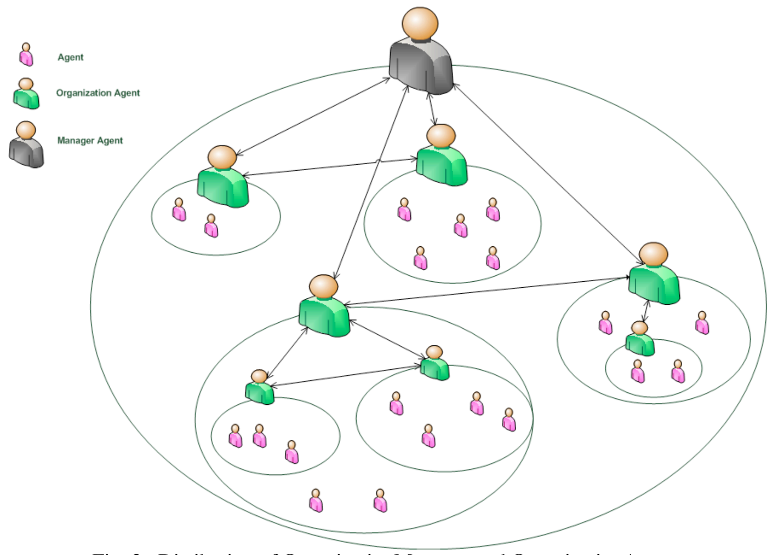


Fig. 2. Distibution of OrganizationManager and OrganizationAgents

Except for the CommunicationAgent (which has its own replication mechanism) and the OrganizationManager (which has the similar OrganizationAgent to avoid overload), the rest of the agents that manage the platform are controlled by the OrganizationManager when faced with the possibility of work overload. Each agent starts with a queue of messages. If the queue is too long, the agent itself will be responsible for instantiating new agents to share the work, thus creating a new organization. The occurs most commonly with the InformationAgent. This agent, who is in charge of data base access, can find itself overloaded with requests. Using a duplication mechanism, it can create instances of other InformationAgents although they would each be dependent on the initial InformationAgent. The newly created InformationAgents do not have their own task queue; instead the parent InformationAgent is in charge of managing a single task queue that it assigne to the children InformationAgents according to the task distribution mechanism **¡Error! No se encuentra el origen de la referencia.**. In this way, the platform would go from being an InformationAgent to an InformationOrganization; that is, an organization that would carry out tasks originally assigned to a single agent. This would avoid that the overload or duplication of the message queues for each agent.

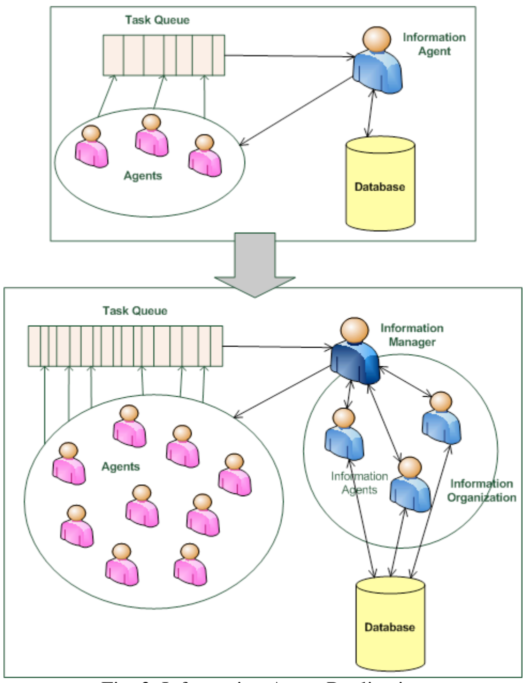


Fig. 3. Information Agent Replication

Communication Platform

This section will focus on describing the communication platform and protocol. PANGEA will not pretend to present a new communication protocol; instead it will introduce the IRC protocol within multi-agent systems. This protocol is widely used in other distributed environments and has already demonstrated its reliability and robustness. What is proposed is its use within the platform, providing advantages, such as ease of implementation and reliability, given that it has been widely used in online communities with good functionality.

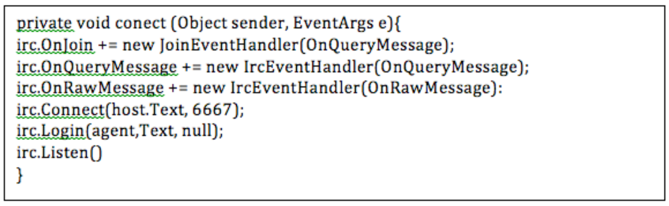
As observed in Figure 1, the communication platform includes two main agents: the CommunciationAgent and the Sniffer. The first is in charge of checking the connections to confirm that the agents are online and see which ones have disconnected. It is also in continual communication with the NormAgent to ensure that the agents respect the lines of communication and comply with the standards. The Sniffer is in charge of recording all communication, offers services so that other agents can obtain history information, and facilitates the control of information flow for programmers and users.

The IRC protocol was used to implement communication. Internet Relay Chat (IRC) is a real time internet protocol for simultaneous text messaging or conferencing. This protocol is regulated by 5 standards [OIKARINEN, J. 2000] [KALT, C. 2000]: RFC1459, RFC2810, RFC2811, RFC2812 y RFC2813. It is designed primarily for group conversations in discussion forums and channel calls, but also allows private messaging for one on one communications, and data transfers, including file exchanges. The protocol in the OSI model is located on the application layer and uses TCP or alternatively TLS. An IRC server can connect with other IRC servers to expand the user network. Users access the IRC networks by connecting a client to a server. There have been many implementations of clients, including mIRC or XChat. The original protocol is based on flat text (although it was subsequently expanded), and used TCP port 6667 as its primary port, or other nearby ports (for example TCP ports 6660-6669, 7000). The standard structure for an IRC server network is a tree configuration. The messages are routed only through those nodes that are strictly necessary; however, the network status is sent to all servers. When a message must be sent to multiple recipients, it is sent to a multidiffusion; that is, each message is sent to a network link only once. This is a strong point in its favor compared to the no-multicast protocols such as SimpleMail Transfer Protocol (SMTP) or the Extensible Messaging and Presence Protocol (XMPP).

One of the most important features that characterize the platform is the use of the IRC protocol for communication among agents. This allows for the use of a protocol that is easy to implement, flexible and robust. The open standard protocol enables its continuous evolution. There are also IRC clients for all operating systems, including mobile devices.

All messages include the following format: *prefix command command-parameters\r\n*. The prefix may be optional in some messages, and required only for entering messages; the command is one of the originals from the IRC standard.

Another advantage in using IRC involves the ease in implementing communication. The platform's code generating tool makes it possible to easily create an outline of an agent, with the communication code requiring few lines of code. Figure 4 displays the code for an agent in C#. It is clear that the functionality of the code consists of associating different events to the *OnqueryMessage* method, intercepting when an agent receives a message or enters an organization, and effectively handling that action from the *OnqueryMessage* method. The Connect method specifies the host and the communication server port, which is responsible for enabling all agents to connect and communicate. The *OnRawMessage* event is responsible for intercepting all server responses.



```
private void conect (Object sender, EventArgs e){
irc.OnJoin += new JoinEventHandler(OnQueryMessage);
irc.OnQueryMessage += new IrcEventHandler(OnQueryMessage);
irc.OnRawMessage += new IrcEventHandler(OnRawMessage):
irc.Connect(host.Text, 6667);
irc.Login(agent,Text, null);
irc.Listen()
}
```

Fig. 4. Example of the communication implementation.

4 Case Study

This platform was used to build the base architecture of an integral support systems for disabled persons in the workplace. Due to its service orientation, different tools modeled with agents that consume Web services can be integrated and operated from the platform, regardless of their physical location or implementation. More specifically, the developed system aims to facilitate the employment of disabled people. We will now see how the different tools are deployed on the platform and are managed by the agents that compose it.

For this case study, different tools were developed using the platform, which facilitates their subsequent integration: the proximity detection tool, which works directly with the customization tool; and the translation tool made of services to communicate with Morse code and through videos. We will briefly present each one. Each of the tools was designed by different teams but using PANGEA as a basis for work. As a result, an integrated system was easily obtained.

The agents involved in the case study are deployed in a specialized suborganization inside PANGEA, each one of this agents offer services (like a localization service) modeled as Web Services. The platform agents are implemented with Java, although the prototype detection agents are implemented in .NET and nesC. Every disabled user in the proposed system carries a Zigbee tag, which is detected by a ZigBeeReaderAgent located in each system terminal and continuously in communication with the ClientComputerAgent. Thus, when a user tag is sufficiently close to a specific terminal (within a range defined according to the strength of the signal), the ZigBeeReaderAgent can detect the user tag and immediately send a message to the ClientComputerAgent, which is coordinated by the ZigBeeCoordinatorAgent. The system uses a LAN infrastructure that uses the wake-on-LAN protocol for the remote switching on and off of equipment.

This tool works together with the customization tool, also displayed as a suborganization within the PANGEA platform. The detection and identification of a user makes it possible to detect any special needs, and for the computer to be automatically adapted for individual use. This allows the system to define and manage the different profiles of people with disabilities, facilitating their job assimilation by automatically switching on or off the computer upon detecting the user's presence, or initiating a procedure that automatically adapts the computer to the personal needs of the user.

The agents involved are:

- ZigbeeManagerAgent: manages communication and events and is deployed in the server machine.
- ClientComputerAgent: user agents located in the client computer and responsible for detecting the user's presence with ZigBee technology, and for sending the user's identification to the ZigbeeManagerAgent. These agents are responsible for requesting the profile role adapted for the user to the ProfileManagerAgent in the ProfileOrganization.
- ZigBeeCoordinatorAgent: an agent included in a ZigBee device responsible for coordinating the other ZigBee devices in the office. It is connected to the server by a serial port, and receives signals from each of the ZigBee tags in the system.
- ZigBeeReaderAgent: these agents are included in several ZigBee devices that are used to detect the presence of a user. Each ZigBeeReaderAgent is located in a piece of office equipment (computer).
- ProfileManagerAgent: responsible for managing user profiles. These agents can receive requests from the ClientComputerAgent.
- DatabaseAgent: stores data related to the users, sensors, computer equipment and user profiles. It can also communicate with the InformationAgent of PANGEA to obtain general information.
- InterfaceAgent: allows the user to set up a personal profile for the first time, moreover, it stays aware of the configuration changes that can occur during the execution and store them for the next execution.

The translation tool emerged as a result of the difficulty encountered by employers in communicating to their hearing impaired employees the actions that they have to do in their jobs. Given the ineffectiveness of avatar translators, the solution chosen was to study the main communication needs and provide some recorded videos with commands and explanations specifically related to the performance of a particular job. These videos are pre-recorded by a sign language interpreter and stored on a Web server where they can be accessed anytime through the request of an issuing agent. The issuing agents, deployed on both Smartphones (Android or iPhone) or computers, will be responsible for playing the video required at that moment. Receptor agents, also available for SmartPhones or computers, will be responsible for capturing by text or by voice, the command or instruction that the employer wishes to transmit to the disabled employee.

In the platform, the translator agent, which is called VideoTranslatorAgent, is deployed. It is responsible for receiving the instruction and mapping the specific video used by the emitter agent who is requesting the transfer.

The translator agent is deployed within the suborganization TranslatorOrganization. Within this organization, a translation tool designed for people with visual disabilities is also displayed. The tool consists of a vibrating bracelet that receives impulses to transmit messages in Morse code. As with the previous system, the employer by a mobile agent or an agent deployed on his computer sends a text, which receives a second translator agent, MorseTranslatorAgent. This agent interprets the message by translating it into Morse code and then uses Bluetooth to send the message content to the the bracelet. The bracelet transmits vibrations to the disabled worker, who is the receiver of the message.

Finally, for monitoring purposes all workers are represented by DisabledAgents and are adhered to the suborganization OfficeOrganization when the proximity detection tool detects them in a workplace.

In summary, the suborganizations in were developed in this case study within the PANGEA platform. For simplicity, interactions between OrganizationAgents are not explicitly drawn:

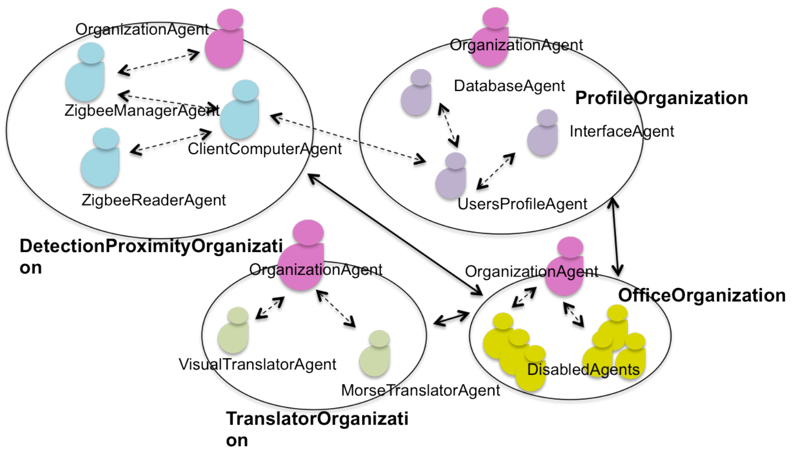


Fig. 5. Suborganizations deployed in PANGEA

As general behavior, agents of a suborganization cannot communicate with other agents of a different suborganization. In this case, using to the NormAgent, which ensures that all platform requirements are fulfilled, a norm was modelled to allow communication between the ProfileManagerAgent of the ProfileOrganization and the ClientComputerAgent of the DetectionProximityOrganization. The NormAgent makes it possible to modify the behavior of certain agents by prohibiting or allowing the consumption of certain services or cutting off communication with other agents.
Moreover, the ServiceAgent agent is also essential in the development of this integrated system, since it records all the services offered by each agent and the address for the invocation of those services. Furthermore, it provides service discovery mechanisms, in case consumer agents need a service location.

5 Conclusions

PANGEA is a complete and innovative platform. We can conclude that PANGEA has great potential to create open multi-agent systems, and more specifically, virtual agent organizations. One of the greatest advantages of this system is the communication platform that, by using the IRC standard, offers a robust and widely tested communication system that can handle a large number of connections and ensure scalability.
This protocol also offers reliability. In the tests carried out, the deployed agents were able to send and receive all messages without losses. Furthermore, the use of the Communication and Sniffer agents offers services that can be easily invoked to study and extract message information.
Another reason that justifies the scalability of the platform is the way of modelling the services as SOA architecture compliant and using Web Services.
The platform offers an IDE, which facilitates the implementation process. It automatically offers the skeleton of an agent and the communication between agents can be implemented with few lines of code.
Finally, the platform admits mobile agents and agents in any programming language, it is not necessary to learn a new language in order to use it.

6 Acknowledgment

This work has been supported by the Spanish Ministry of Science and Innovation project “Sociedades Humano-Agente: inmersión, adaptación y simulación (iHAS) TIN2012-36586-C03-03

7 References

[FERBER, J. *et al*. 2008] J. Ferber, O. Gutknecht, F. Michel, From Agents to Organizations: an Organizational View of Multi-Agent Systems, in: P. Giorgini, J. Muller, J. Odell (Eds.), Agent-Oriented Software Engineering VI, LNCS Springer-Verlag. 2935, pp. 214–230.

[FOSTER, I. *et al*. 2001] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, Int. J. High Perform. Comput. Appl. 15 (3) (2001) 200-222

[BAJO, J. *et al*. 2009] J. Bajo, J.M. Corchado, V. Botti, S. Ossowski. Practical applications of agents and MAS: methods, techniques and tools for open MAS. Journal of Physical Agents, 3 (1–2) 2009.

[HELSINGER, A. *et al*. 2004] A. Helsinger, M. Thome, T. Wright, Cougaar: a scalable, distributed multi-agent architecture, Systems, Man and Cybernetics, 2004 IEEE International Conference on, 2, (2004) pp. 1910- 1917.

[GRUVER, A. 2004] W. Gruver, Technologies and Applications of Distributed Intelligent Systems, IEEE MTTChapter Presentation, Waterloo, Canada, 2004.

[EMORPHIA, 2013] Emorphia, FIPA-OS. http://fipa-os.sourceforge.net/ last access 30/08/2013

[POSLAD, S. *et al*. 2000] S. Poslad, P. Buckle.R. Hadingham, The FIPA-OS agent platform: Open Source for Open Standards. In Procedings of Autonous Agents AGENTS-2000, Barcelona, 2000.

[MCCABE, A. 1995] F.G. McCabe, K.L. Clark. APRIL—Agent PRocess Interaction Language. In Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents (ECAI-94), Michael J. Wooldridge and Nicholas R. Jennings (Eds.). Springer-Verlag New York, Inc., New York, NY, USA, 1995, 324-340.

[BORDINI, R. *et al*. 2005] R.H. Bordini, J.F. Hübner, R. Vieira. Jason and the Golden Fleece of agent-oriented programming. In Bordini, R. H., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., eds., Multi-Agent Programming: Languages, Platforms and Applications. Springer-Verlag. chapter 1, 2005, pp. 3-37.

[JASON, 2013] JASON. http://jason.sourceforge.net/Jason/Jason.html last access 30/08/2013

[BORDINI, R. *et al*. 2007] R.H. Bordini, J.F. Hübner, M. Wooldridge. Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd. 2007.

[CORCHADO, J. *et al*. 2008] J.M. Corchado, M. Gonzalez-Bedia, Y. De Paz, J. Bajo, J.F. De Paz. Replanning mechanism for deliberative agents in dynamic changing environments. Computational Intelligence, 24 (2) (2008) 77-101

[RAO, A. *et al*. 1991] A.S. Rao, M.P. Georgeff. Modeling rational agents within a BDI-Architecture. In J. Allen, R. Fikes, & E. Sandewall (Ed.), Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning ({KR}'91). San Mateo, CA, USA: Morgan Kaufmann publishers, Inc. 1991, pp. 473-484.

[BELLIFEMINE, F. *et al*. 1999] F. Bellifemine, A. Poggi and G. Rimassa. JADE – A FIPA-compliant agent framework. Proceedings of the Practical Applications of Intelligent Agents, 1999.

[POKAHR, F. *et al*. 2005] A. Pokahr, L. Braubach, W. Lamersdorf. A BDI Reasoning Engine. Multi-Agent Programming. Systems, Artificial Societies, and Simulated Organizations, Springer US , 15 2005, pp.149-174.

[HIRSCH, B. *et al*. 2009] B. Hirsch, T. Konnerth, A. Heßler, Merging Agents and Services — the JIAC Agent Platform. Multi-Agent Programming. El Fallah Seghrouchni, A., Dix, J., Dastani, M., Bordini, R. (Eds). Springer US, 2009, pp.159 – 185.

[LUTZENBERGER, M. *et al*. 2009] M. Lutzenberger, B. Hirsch, T. Konnerth, A, Heßler, Unifying JIAC Agent Development with AWE. MATES'09 Proceedings of the 7th German conference on Multiagent system technologies, 2009, pp. 220-225.

[GUTKNECHT, O. *et al*. 2000]	O. Gutknecht, J. Ferber. The MadKit Agent Platform Architecture. In Proccedings Agents Workshop on Infrastructure for Systems 2000, pp. 48-55.
[GUTKNECHT, O. *et al*. 1997]	O. Gutknecht, J. Ferber. MadKit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical Report R.R.LIRMM 9718, LIRM, 1997.
[JACK, 2005]	Agent Oriented Software Pty Ltd. JACK™ Intelligent Agents Teams Manual. s.l. : Agent Oriented Software Pty Ltd, 2005.
[GIRET, A. 2009]	A. Giret, An open architecture for Service-Oriented Virtual Organizations. Programming Multi-Agent Systems: 7th International Workshop, ProMAS 2009.
[GALLAND, S. 2010]	S. Galland. JANUS: Another Yet General-Purpose Platform. Seventh AOSE Technical Forum, Paris 2010.
[JARVIS, J. *et al*. 2006]	J. Jarvis, R. Rönnquist, D. McFarlane, L. Jain. A team-based holonic approach to robotic assembly cell control. Journal Network and Computer Applications. 29 (2), 2006, pp. 160-176.
[BISHT, A. *et al*. 2007]	S. Bisht, A. Malhotra, S.B. Taneja. Modelling and Simulation of Tactical Team Behaviour. Defence Science Journal. 57 (6) 2007 pp. 853-864.
[GAUD, N. *et al*. 2008]	N. Gaud, S. Galland, V. Hilaire. A. Koukam. An Organizational Platform for Holonic and Systems. In Proccedings of Sixth International Workshop on Programming Multi-Agent Systems (ProMAS'08), of the Seventh International Conference on Autonomous agents and Systems (AAMAS). E. Hindriks, A. Pokahr and S. Sardina (Eds.), 2008, pp. 111–126.
[HUBNER, J. *et al*. 2002]	J.F. Hubner, J.S. Sichman, O. Boissier. A model for the structural, functional, and deontic specification of organizations in systems. In Guilherme Bittencourt and Geber L. Ramalho, editors, Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), LNAI Springer- Verlag, 2507,2002, pp.118-128.
[HUBNER, J. *et al*. 2006]	J.F. Hubner, J.S. Sichman, O. Boissier. S-MOISE+: A middleware for developing organised multi-agent systems. In Olivier Boissier, Virginia Dignum, Eric Matson, and Jaime Simao Sichman, editors, Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, LNAI Springer- Verlag, 3913, 2006, pp. 64-78
[HUBNER, J. *et al*. 2009]	Hübner, J.F., Bordini, R.H., Picard, G.: Using Jason and MOISE+ to develop a team of cowboys. In: Hindriks, K., Pokahr, A., Sardina, S. (eds.) Proceedings of the Seventh International Workshop on Programming Multi-Agent Systems (ProMAS 08), Agent Contest, held with The Seventh International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2008), LNAI, vol. 5442, pp. 238–242. Springer, Heidelberg (2009)
[HUBNER, J. 2006]	J.F. Hübner. J -Moise+ Programming organisational agents with Moise+ & Jason. Technical Fora Group at EUMAS'07.
[SIERRA, C. *et al*. 2004]	C. Sierra, J.A. Rodríguez-Aguilar, P. Noriega, M. Esteva, J.L Arcos. Engineering multi-agent systems as electronic institutions in European Journal for the Informatics Professional, V(4), 2004, pp. 33-39.
[COSSENTINO, M. *et al*. 2010]	M. Cossentino, N. Gaud, V. Hilaire, S. Galland, A. Koukam. ASPECS: an Agent-oriented Software Process for Engineering Complex Systems. International Journal of Autonomous Agents and Multi-Agent Systems (IJAAMAS). 20(2). 2010.
[ESTEVA, M. *et al*. 2003]	M. Esteva. Electronic Institutions: from specification to development Ph. D. Thesis, Technical University of Catalonia, 2003.
[JOSUTTIS, N. *et al*. 2007]	N.M. Josuttis, SOA in Practice. O´Reilly Media, Inc. Agosto, 2007
[OIKARINEN, J. 2000]	J. Oikarinen, D. Reed, Internet Relay Chat Protocol, RFC 1459, May 1993.

[KALT, C. 2000]	C. Kalt, Internet Relay Chat: Client Protocol, RFC 2812, April 2000. Internet Relay Chat: Server Protocol, RFC 2813, April 2000. Internet Relay Chat: Channel Management, RFC 2811, April 2000. Internet Relay Chat: Architecture, RFC 2811, April 2000.
[WU, D. *et al*. 2003]	D. Wu, E. Sirin, J. Hendler, D. Nau, B. Parsia. Automatic web services composition using SHOP2. In Workshop on Planning for Web Services, ICAPS, (2003)
[ZATO, C. *et al*. 2012]	C. Zato, J.F. De Paz, A. de Luis, J. Bajo, J.M. Corchado, Model for assigning roles automatically in egovernment virtual organizations, Expert Systems with Applications, 39 (12) (2012) 10389-10401.
[BAJO, J. *et al*. 2010]	J. Bajo, J.A. Fraile, B. Pérez-Lancho, J.M. Corchado, The THOMAS architecture in Home Care scenarios: A case study, Expert Systems with Applications. 37 (5) (2010) 3986-3999.