PANGEA – Platform for Automatic coNstruction of orGanizations of intElligent Agents

Carolina Zato, Gabriel Villarrubia, Alejandro Sánchez, Ignasi Barri, Edgar Rubión, Alicia Fernández, Carlos Rebate, José A. Cabo, Téresa Álamos, Jesús Sanz, Joaquín Seco, Javier Bajo, and Juan M. Corchado

Abstract. This article presents PANGEA, an agent platform to develop open multiagent 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.

Keywords: multiagent platform, Web services, virtual organizations, IRC protocol.

Carolina Zato · Gabriel Villarrubia · Juan M. Corchado
Departamento Informática y Automática, Universidad de Salamanca, Salamanca, Spain
e-mail: {carol_zato,gvg,corchado}@usal.es

Alejandro Sánchez · Javier Bajo
Universidad Pontificia de Salamanca, Salamanca, Spain
e-mail: {asanchezyu,jbajope}@usal.es

Ignasi Barri · Edgar Rubión · Alicia Fernández · Carlos Rebate
Indra, Spain
e-mail: {ibarriv,erubion,afernandezde,crebate}@indra.es

José A. Cabo · Téresa Álamos
Wellness Telecom, Spain
e-mail: {talamos,jacabo}@wtelecom.es

Jesús Sanz · Joaquín Seco
CSA, Spain
e-mail: {jesus.sanz,joaquin.seco}@csa.es

S. Omatu et al. (Eds.): Distributed Computing and Artificial Intelligence, AISC 151, pp. 229–239.
springerlink.com

1 Introduction

One of the current lines of investigation for multiagent systems aims to create an increasingly open and dynamic system. This involves adding new capabilities such as adaption, reorganization, learning, coordination, etc. Virtual agent Organizations (VOs) [1][2] emerged in response to this idea; they include a set of agents with roles and norms that determine their behavior, and represent a place where these new capabilities will assume a critical role. Possible organizational topologies and aspects such as communication and coordination mechanisms determine in large part the flexibility, openness and dynamic nature that a multiagent system can offer.

There are many different platforms available for creating multiagent systems that facilitate the work of the agent; however those that allow for the creation of VOs number much fewer, and it is difficult to find one single platform containing all of the requirements for a VO.

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. Finally, section 4 explains a case study and presents some results.

2 Related Works

All platforms for creating multiagent systems existing 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. We will first present those platforms that do not incorporate organizational aspects. The FIPA-OS [4] agent platform was created as a direct derivative of the FIPA [3] standard. Another agent platform is the April Agent Platform (AAP) [5] which, unlike the majority of platforms using Java, implements the April language [6]; its development and technological support has been discontinued. One of the strong points of this platform is that it provides services to facilitate the development and deployment of agents on the Internet and is also compliant with Web Services and Semantic Web standards.

One of the most recent platforms still in development is JASON [7][8]. Its greatest contribution is the easy implementation of BDI agents [10]. The Java-developed platform contains AgentSpeak in its nucleus, an interpreter agent that acts as a language extension [9]. The platform offers two operation modes: one that runs all agents in the same machine, and another which allows distribution using SACI (Simple Agent Communication Infrastructure) [11], which in turn uses KQML [24] language instead of RIPA-ACL [23]. In practice, the most used platform for developing multiagent systems in real case studies is JADF (Java Agent Development Framework) [12]. The JADE platform focuses on implementing the FIPA reference model, providing the required communication infrastructure and platform services such as agent management, and a set of development and debugging tools. Jadex [13] is a software framework for the

creation of goal-oriented agents following the belief-desire-intention (BDI) model. The Jadex project facilitates a smooth transition from developing conventional JADE agents to employing the mentalistic concepts of Jadex agents.

With the exception of JASON, these platforms follow the FIPA standard, can create agents (some with different models), and manage communication among agents and services. With VOs, however, it is necessary to consider the normative and organizational aspects that the platform itself must provide. MadKit [20] was one of the first platforms to consider basic organizational aspects. The platform architecture is rooted in the AGR (agent-group-role) model [14]; however, while it can handle the concept of role, it does not consider a role a class entity, and the behavior associated with the role is directly implemented in the agent who assumes it. Roles are strongly linked to agent architectures. This approach harms the reusability and modularity of organizations [15].

Another pioneering platform with regards to structural aspects was Jack Teams [16]. JACK Teams is an extension of JACK Intelligent Agents [17], which provides a team-oriented modelling framework. Both are extensions of the Java programming language; the implemented source code is first compiled into regular Java code before being executed.

S-MOISE+ is an organizational middleware that follows the MOISE+ model [18]. It is an extension of SACI [11] where the agents have an organizational aware architecture. Our research found systems developed in conjunction with JASON and using S-Moise+ as middleware to achieve a more complete model [19]. The result was J-Moise+ [20], which is very similar to S-Moise+ regarding overall system concepts. The main difference is how the agents are programmed: in S-Moise+ agents are programmed in Java (using a very simple agent architecture), while in J -Moise+ they are programmed in AgentSpeak.

One of the main disadvantages of VO oriented platforms is the slight loss in the concept of service and, consequently, the management of these services and the Directory Facilitator (DF) described in the FIPA standard. THOMAS was developed in response to this twofold need. 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 wholly independent of any internal agent platform and fully addressed for open multiagent systems [21].

Finally, one of the most complete and recent platforms that we found is Janus [22]. Janus is the next step towards platform organizations known as TinyMAS (no longer under development.). This platform was specifically designed to deal with the implementation and deployment of holonic and multiagent systems. Its primary focus is to support the implementation of the concepts of role and organization as first-class entities (a class in the object-oriented sense). This consideration has a significant impact on agent implementation and allows an agent to easily and dynamically change its behaviour [15].

In conclusion, it could be said that when dealing with all aspects of complex multiagent systems such as VOs, it is also necessary to deal with multiple levels of abstractions and openness, which is not the case for most solutions.

3 Architecture Overview

As we have mentioned, we are looking for a platform that can integrally create, manage and control VOs. In general terms, the proposed platform includes the following characteristics:

- Different models of agents, including a BDI and CBR-BDL architecture.
- 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.
- Allow organizations with any topology.
- Organization management.
- Services for dynamically reorganizing the organization.
- Services for distributing tasks and balancing the workload.
- A business rules engine to ensure compliance with the standards established for the proper operation of the organization.
- Programmed in Java and easily extensible.
- Possibility of having agents in various platforms (Windows, Linux, MaccOS, Android and IOS)
- Interface to oversee the organizations.

Figure 1 displays the principal entities of the system, and illustrates how the roles, norms and the organizations themselves are classes that facilitate the inclusion of organizational aspects. The services are also included as entities completely separate from the agent, facilitating their flexibility and adaption.

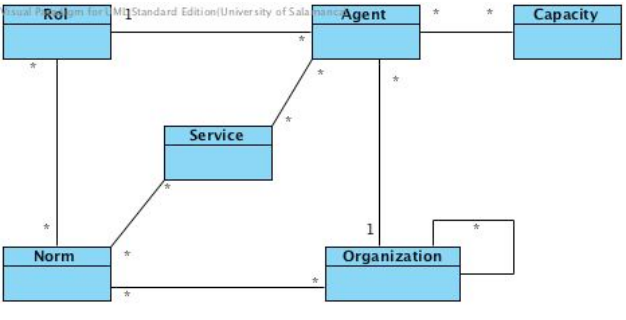

Fig. 1 Principal classes of the system

When launching the main container of execution, the communication system is initiated; the agent platform then automatically provides the following agents 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.

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.

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.

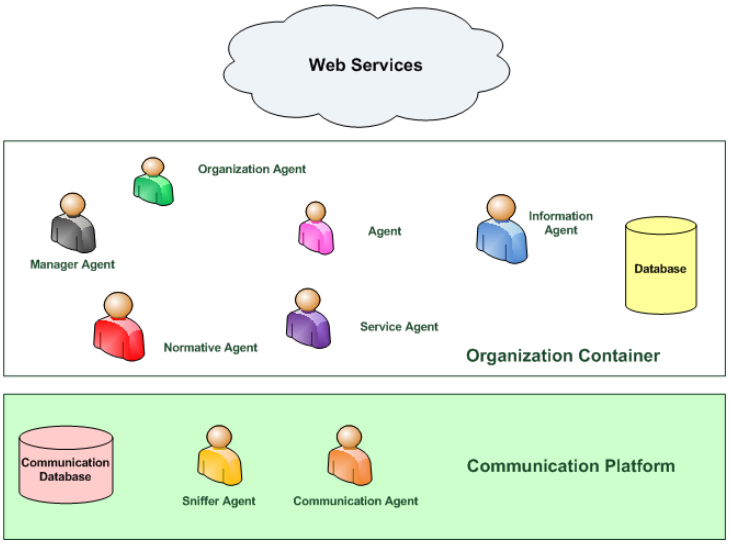

Fig. 2 Architecture

The platform examines 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 hope to create a service oriented platform that can take maximum advantage of the distribution of resources. To this end, all services are implemented 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 knows how to contact the web service; once the client agent's request has been received, the provider agent extracts the required parameters and establishes the contact. Once received, the results are sent to the client agent.

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.

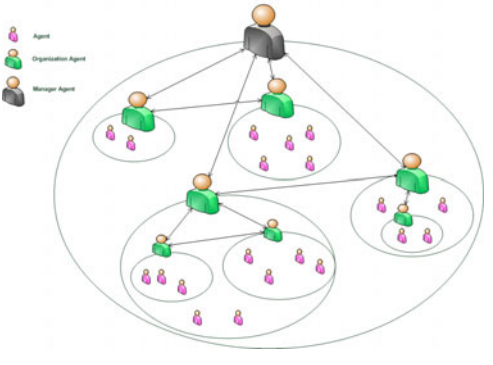

Fig. 3 OrganizationManager and OrganizationAgents

3.1 Communication Platform

This section will focus on describing the communication platform and protocol. As observed in Figure x, 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: RFC1459 [25], RFC2810 [29], RFC2811 [28], RFC2812 [26] y RFC2813 [27]. 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 [25]. The protocol in the OSI model is located on the application layer and uses TCP or alternatively TLS [29]. 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) [26]. 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 similar to a multidiffusion; that is, each message is sent to a network link only once [29]. 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.

The following diagram illustrates the message flow required for an agent to enter an organization. These messages use the command PRIVMSG followed by the parameters indicated by the arrows in the diagram.

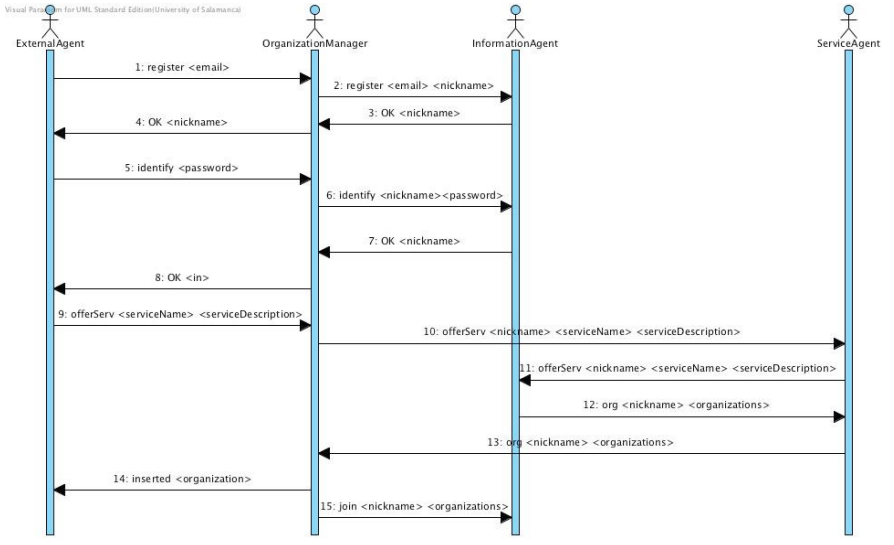

Fig. 4 Sequence of steps for an agent to enter an organization

4 Case Study and Results

The platform we have developed can create a general type of organization, and includes the possibility of creating open and highly dynamic systems. In order to test the architecture, 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 one possible case, 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. If the message contains the keyword “invoice”, the query will be handled by the Accounting agent; if the keyword is “switch on” it will be handled by the Technical agent. Once the client is in contact with the appropriate agent, the agent can communicate with other agents in its organization to carry out the task.

Four 30-minute simulations were performed with 20 different types of requests randomly provided. Studying the Evaluation and Sniffer agents it was possible to study how both the simulation and message flow unfolded. Focusing specifically on the Sniffer, it is possible obtain summary charts and diagrams, and specific numbers. Once the query is made, the Sniffer consults the database, filters the data and returns a URL that displays the desired data.

It is possible to 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.

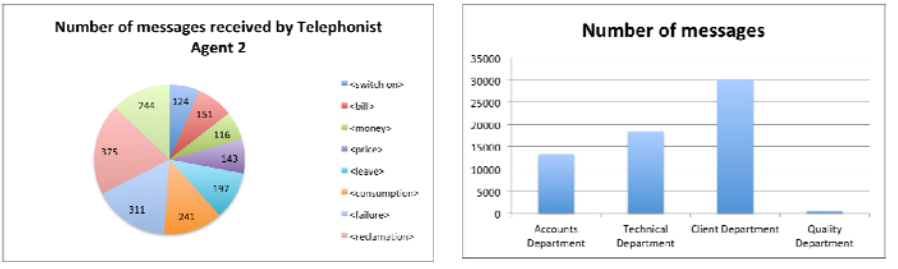

Fig. 5 Diagramas of messages.

It is also possible to obtain a diagram of messages according to organization instead of agents. 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.

We can conclude that the architecture we are developing has great potential to create open systems, and more specifically, virtual agent organizations. This architecture includes various tools that make it easy for the end user to create, manage and control these systems. One of the greatest advantages of this system 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, and that additionally facilitates the implementation for other potential extensions. Furthermore, the use of the Communication and Sniffer agents, offers services that can be easily invoked to study and extract message information.

Acknowledgements. This project has been supported by the Spanish CDTI. Proyecto de Cooperación Interempresas. IDI-20110343, IDI-20110344, IDI-20110345, and the MICINN TIN 2009-13839-C03-03 project. Project supported by FEDER funds.

References

1. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: An Organizational View of Multi-Agent Systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
2. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. 15(3), 200–222 (2001)
3. O'Brien, P.D., Nicol, R.C.: FIPA, Towards a Standard for Software Agents. BT Technology Journal 13(3), 51–59 (1998)
4. Emorphia, FIPA-OS, `http://fipa-os.sourceforge.net/`

5. Dale, J., Knottenbelt, J., Labo, F.: April Agent Platform (2011), http://designstudio.lookin.at/research/relate%20survey/Survey%20Agent%20Platform/April%20Agent%20Platform.htm (accessed November 29, 2011)
6. McCabe, F.G., Clark, K.L.: APRIL—Agent PRocess Interaction Language. In: Wooldridge, M.J., Jennings, N.R. (eds.) Proceedings of the Workshop on Agent Theories, Architectures, and Languages on Intelligent Agents (ECAI 1994), pp. 324–340. Springer, New York (1995)
7. Bordini, R.H., Hübner, J.F., Vieira, R.: Jason and the Golden Fleece of agent-oriented programming. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, ch. 1, pp. 3–37. Springer (2005)
8. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in Agent Speak Using Jason. John Wiley & Sons, Ltd. (2007)
9. Rao, A.S.: Agent Speak(L): BDI agents speak out in a logical computable language. In: Perram, J., Van de Velde, W. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
10. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-Architecture. In: Allen, J., Fikes, R., Sandewall, E. (eds.) Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 1991), pp. 473–484. Morgan Kaufmann publishers, Inc., San Francisco (1991)
11. SACI – Simple Agent Communication Infrastructure, http://www.lti.pcs.usp.br/saci/
12. Bellifemine, F., Poggi, A., Rimassa, G.: JADE – A FIPA-compliant agent framework. In: Proceedings of the Practical Applications of Intelligent Agents (1999)
13. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A Short Overview. Procceding Main Conference Net. Object Days, 195–207 (2004)
14. Gutknecht, O., Ferber, J.: MadKit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical Report R.R.LIRMM 9718, LIRM (December 1997)
15. Gaud, N., Galland, S., Hilaire, V., Koukam, A.: An Organisational Platform for Holonic and Multiagent Systems. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS, vol. 5442, pp. 104–119. Springer, Heidelberg (2009)
16. Agent Oriented Software Pty Ltd. JACK^{TM} Intelligent Agents Teams Manual. s.l. : Agent Oriented Software Pty Ltd. (2005)
17. Busetta, P., Rönnquist, R., Hodgson, A., Lucas, A.: JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report. Agent Oriented Software Pty. Ltd., Melbourne, Australia (1998)
18. Hubner, J.F., Sichman, J.S., Boissier, O.: A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In: Bittencourt, G., Ramalho, G.L. (eds.) SBIA 2002. LNCS (LNAI), vol. 2507, pp. 118–128. Springer, Heidelberg (2002)
19. Hübner, J.F., Bordini, R.H., Picard, G.: Using Jason and MOISE+ to Develop a Team of Cowboys. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS, vol. 5442, pp. 238–242. Springer, Heidelberg (2009)
20. Hübner, J.F.: J -Moise+ Programming organisational agents with Moise+ & Jason. Technical Fora Group at EUMAS 2007 (2007)

21. Giret, A., Julián, V., Rebollo, M., Argente, E., Carrascosa, C., Botti, V.: An Open Architecture for Service-Oriented Virtual Organizations. In: Braubach, L., Briot, J.-P., Thangarajah, J. (eds.) ProMAS 2009. LNCS, vol. 5919, pp. 118–132. Springer, Heidelberg (2010)
22. Galland, S.: JANUS: Another Yet General-Purpose Multiagent Platform. Seventh AOSE Technical Forum, Paris (2010)
23. The Foundation for Intelligent Physical Agents. FIPA Standar Status Specification (2011), http://www.fipa.org/repository/standardspecs.html (accessed November 11, 2011)
24. Finin, T., Labrou, Y.: KQML as an agent communication language. In: Bradshaw, J.M. (ed.) Software Agents, Cambridge, MA, pp. 291–316 (1997)
25. Oikarinen, J., Reed, D.: Internet Relay Chat Protocol. RFC 1459 (May 1993)
26. Kalt, C.: Internet Relay Chat: Client Protocol. RFC 2812 (April 2000)
27. Kalt, C.: Internet Relay Chat: Server Protocol. RFC 2813 (April 2000)
28. Kalt, C.: Internet Relay Chat: Channel Management. RFC 2811 (April 2000)
29. Kalt, C.: Internet Relay Chat: Architecture. RFC 2811 (April 2000)