CONNECTING JADE WITH PN AGENT

Jakub Žák, Vladimír Janoušek, Radek Kočí, František Zbořil ml.

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech republic

izakjakub@fit.vutbr.cz (Jakub Žák)

Abstract

Since Foundation of Intelligent Physical Agent (FIPA) is one of the biggest organizations that handles standardization in the field of agent-based technologies we have chosen their reference implementation called JADE and in this paper we are describing possibility to interconnect it with other platform that is not FIPA compliant. Second agent platform, we have chosen, is PN agent platform, which is BDI agent based on formalism of Object Oriented Petri nets. PN agent was chosen for the reason of Object Oriented Petri nets used in its base because when we make PN agent FIPA compliant we open whole new world of possibilities how to use Petri nets inside of it.

Main purpose of this paper is to show how to interconnect two different platforms where one is build upon FIPA standards and the other is not. This scenario is hardened by the fact, that JADE is developed in Java language, while PN agent framework is created in smalltalk, concretely in its squeak implementation.

Other topic of our interest is FIPA compliance in agent platforms. This topic partially covers our effort to bring more openness to the PN agent platform because when we find out, which agent platforms support FIPA standards then we will acquire knowledge of who are we compatible with.

Keywords: Agent platform, Connecting agent platforms,Object oriented Petri Nets, PN agent, JADE, FIPA, FIPA compliance.

Presenting Author's biography

Leading author is student of PhD study program on Faculty of Information Technology on Brno University of Technology. He is studying his first year of his PhD studies and this is his first paper. Among his interests belongs all related to agent oriented technologies and modeling systems.

1 Introduction

Nowadays, enterprise developers deal with the problem of constructing large and very complex applications. These applications can be worldwide and must operate across continents or companies, which means, that they work in heterogeneous and very dynamic environment. There is often the requirement to run more instances, which must cooperate with each other. Developing of such application can be so difficult that it can be unsolvable with classical software engineering techniques because of dynamically changing and unmanaged environments. Well suited example of such an environment is the Internet, which is rapidly spreading from its start and becomes so large and wide network never created before. At this point agent-based computation seems to be one promising technology for the development of distributed, open and intelligent systems. First important feature is that agent is an autonomous unit, which can make it's own decisions and plans to achieve some goals. This capabilities determines agents to operate in such dynamic environments. Secondly, multi-agent system (MAS) is inherently parallel so it can be distributed all over the world and agents can live (physically run) in any instance of the application. This implies that one of the key features within MAS is communication.

At first agent systems were designed with ad-hoc solutions (communication, language, architecture and so on) to achieve some specific functionality of an application [1]. It is obvious that this approach isn't very effective, because reusability of such agent system in slightly different scenario is poor. Massive deployment of multi-agent systems, which are built each more different than the other, addresses even more issues. If every solitude system provides some functionality and each system has its own language, communication protocols etc. than they cannot communicate effectively if ever. This gives us many islands of functionality in the sea of rapidly spreading Internet, which can operate only with themselves. If we realize this, we must see that the whole idea of openness and interconnection is teared apart. Last but not least is the view of programmer as a creator of MAS. Many agent frameworks/platforms/toolkits imply many techniques and comprehensive knowledge of every one programmer works with.

Agent-based computation in general is young paradigm and there is no consolidation in software engineering methods for agent oriented technologies. As said in [2]:"*Agent-based technologies cannot realize their full potential, and will not become widespread, until standards to support agent interoperability are available and used by agent developers and adequate environments for the development of agent systems are available.*" This implies the need for some standardization in the area of agent systems. Several researches has been done for this topic and few standards has been founded. We can see for example Knowledge Sharing Effort [3], OMG [4] and FIPA [5]. For our work we have chosen FIPA standards, because only FIPA provides specifications for whole agent-based system and solely for agent-based system. It also has reference implementation called JADE [11].

In the next chapter we introduce main goals of our work, which is basically interconnecting JADE and PN agent [10]. PN agent is agent framework built on Object Oriented Petri net (OOPN) paradigm and programmed in smalltalk. Further we describe some basic ideas, which led us to start this effort. Following chapter includes some basic information about FIPA necessary to understand principles of developed standards. In fourth chapter are briefly outlined some FIPA compliant platforms including JADE with aim on their communication subsystems. Fifth chapter describes possibilities of interconnecting platforms and solution implemented.

2 Aims of the work

As said before, our primary objective is to interconnect PN agent with JADE. By that we enable PN agent to communicate with other agents but we can also let it to exploit services of other agents or parts of platform. Using other agent services can be useful in at least two scenarios. First of them is very prosaic. We can certainly imagine scenario where we are developing an agent or rather agent system for some purpose [1,6,12,13]. In this scenario we are creating agents and we want to be able to debug their conversations somehow. PN agent platform is built upon Object Oriented Petri Net paradigm, which gives it strong mathematical background and possibility to verify agent models, nevertheless it does not have communication debugging tool providing easy control of conversations flow. If we realize that JADE has this tool called sniffer agent we can easily imagine a situation where we connect our communicating agents to JADE and sniff their conversation with sniffer agent. Second scenario assumes similar usage but not with the platform defined tools (agents). In this case we can count on any agent created by JADE users and we can exploit services provided by it (for instance see [12]).

To facilitate interconnection between platforms there is necessity to create interface between JADE platform and PN agent. It should be noted that by interface we don't mean set of method headers (as in Java) but we mean interface as a set of functions that build functionality of gateway between two different platforms.

PN agent is software framework for creating agents based on BDI software model and it is implemented in OOPN by use of smalltalk language concretely its squeak implementation. JADE on the other side is implemented solely in Java language, which brings the obstacle of two different languages. Existence of this obstacle presents even bigger challenge than interconnecting platforms written in the same language. To face this challenge successfully is a contribution for other scientists who are facing similar difficulties.

Other topic we have decided to examine is various platforms compliance with FIPA standards. Since we work with FIPA specifications it is helpful to know what platforms are FIPA compatible and how often they are used in creating agent systems. It is not the intention of this paper to gabble about all agent frameworks ever created but some of them, which we find interesting, will be discussed.

By this knowledge we obtain useful informations about our compatibility to described platforms. This means that if we create some sort of gateway between JADE and PN agent we get compatibility not to JADE only but to FIPA compliant platforms as well.

Last topic of our interest is modeling nodes of wireless sensor network. We have fully operational agent environment [14] and agent language [15] both created for purposes of execution in wireless sensor network. These "wireless agents" can be modeled with use of PN agent so we acquire advantages of formal mathematical apparatus in agent environment.

3 FIPA

The Foundation of Intelligent Physical Agents [5] is a non-profit international association of companies and organizations, which share the effort in the field of standardization in agent oriented technologies. FIPA standards aren't just set of rules for a single application domain but rather a set of general technologies, which support different areas in agent-based systems. The main effort here is to bring interoperability across MAS applications.

To be minimally FIPA compliant, platform must implement at least Agent Management System and Agent Communication Language. Nevertheless FIPA compliance is defined on more levels because founders of FIPA realize that there is sometimes no need to implement all functionality of all standards. To be fully FIPA compliant there is necessity to implement all specifications with mandatory parts for internal and external platform behavior [19].

FIPA standards are based on two main thoughts. The first is that the time to reach consensus in creation standard should not be long and secondly, it should be specified only external behavior of system components, which leaves developers open path in creating insides of their platform components.

In the bottom-up approach FIPA specifications can be divided as follows [1]:

1. *Agent platform* (AP), which presents an infrastructure where agents perform their operations.
2. *Agent communication language* (ACL), that is used to encode messages agents exchange. ACL is based on Speech act theory [7]. By this theory messages are *communicative acts*, which means that there suppose to be an action done or mental attitude changed by the message.
3. *Content language*, which is the language to encode *content* of the sent message. The content is a domain knowledge represented by a content language.
4. *Protocols* are patterns of message exchange. They provide standardization in the flow of conversations among agents.

The FIPA agent reference model (Fig. 1) provide framework, which supports agent existence, messaging capabilities and operability. In connection with agent life cycle management, it creates contexts for creation, operation and retirement of agents.

FIPA specifications follow some inspiring principles that are guiding the standardization process. Among others belongs explicitness, openness and interoperability.

Openness manifests ability to join and leave platform by any agent at run time without any need of reconfiguring or even restart platform. In close relation with openness is interoperability. FIPA tends to specify minimum set of requirements needed for definition of platform in order to avoid any commitments with specific hardware, operating system or programming language.

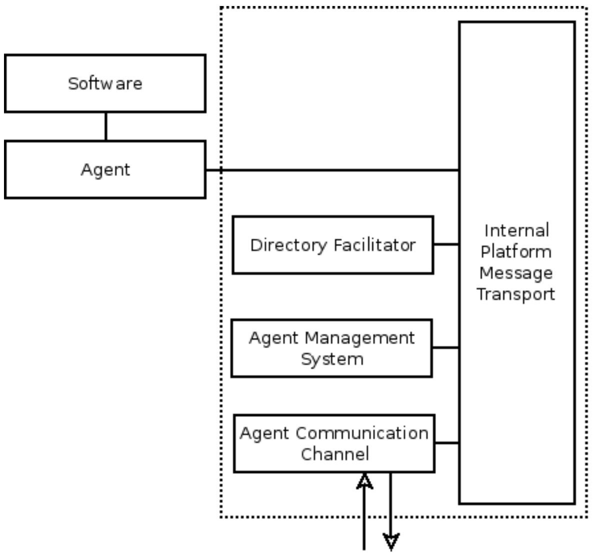
Software
Agent
Directory Facilitator
Agent Management System
Agent Communication Channel
Internal Platform Message Transport

Fig. 1: FIPA agent reference model

Explicitness means, that every information or assumption about agent system including agent roles, capabilities or ontologies should be as explicit as possible. FIPA provides some featured agents to accomplish this. They are depicted in Fig. 1. First of them is Agent Management System (AMS). This agent is supervising access and use of the platform. It is responsible for authentication of resident agents and controls registrations within platform. FIPA standards are speaking in terms of *services* and AMS is providing service called *white pages*, which is a list of agents in platform. Second depicted agent is directory facilitator (DF), which is agent that holds service called *yellow pages*. Yellow pages are in fact register, where ordinary agents can expose their services in order to be exploited by others ordinary agents. Agent Communication Channel (ACC) provides the infrastructure for messaging inside and outside the agent platform.

In FIPA standards there is mandatory support for Internet Inter-Orb Protocol (IIOP), which is implementation of General Inter-Orb Protocol (GIOP) for TCP/IP. GIOP is the abstract protocol founded by OMG [4]. FIPA demands support of IIOP for interoperability with other compliant agent platforms.

As claimed before, FIPA created specifications for agent communication via ACL. Agent communication is based on passing messages to each other. The ACL sets out rules for encoding, semantics and pragmatics of the messages and in the spirit of openness it says nothing about mechanism of transporting messages, because of possibility to use various hardware platforms or network technologies. The syntax of ACL is very close to the syntax of widely used Knowledge Query and Manipulation Language [8]. Most evident difference between these two is existence of formal semantics for FIPA ACL, which should eliminate any misinterpretation in agent messaging. There are some interesting features on FIPA ACL that are worth to mention:

- It is independent on actual message content, since it defines only communicative intention of transferred message.
- Its semantics enables agent to consider a message in an explicit manner, which means that a communicative act can be planned as a normal action.
- ACL provides bases for specification of interaction protocols and common patterns of conversations between agents.

FIPA supports commonly used protocols to widen its interoperability possibilities. These range from simpler ones (such as simple query and request protocols) to the more complex ones where belongs Contract Net interaction protocol [9] or well known English and Dutch auctions.

The remainder of FIPA specifications is focused on the other aspects of agent system, in particular with agent-software integration, agent security, agent mobility, ontology service and others. Since we are interested in interconnecting platforms we need to work closely with communication aspects of standards and that is why informations about FIPA provided here are mainly focused on this area.

4 FIPA compliant agent platforms

Throughout years lot of researches has been done in developing agent platforms, frameworks and toolkits and a lot of them were produced. This chapter aims at outlining some them and their FIPA compliance. After reading this article one should acknowledge that FIPA is if not biggest than at least major player in the field of agent-based systems standardization. For the full list of major publicly implementations of agent platforms which conform to the FIPA Specifications see [16]. In description of chosen platforms we will focus mainly on their communication subsystem to outline interconnection possibilities.

Among compliant platforms is also JADE which communication subsystem will be described more precisely to present proper information, which we can our later effort built upon.

4.1 Zeus

Zeus [17] is the toolkit for creating deliberative agent systems. It follows principles that makes it generic, customizable and scalable. Among those principles belongs delineation, domain-level problem solving capabilities and agent-level functionality, support for open design to ensure extensibility and use of standards wherever possible. As an example of standard we can name use of KQML communication language.

Agent in Zeus toolkit is an entity, which architecture is divided into layers. There are API Layer, Definition Layer, Organization Layer, Coordination Layer and Communication Layer. Since we are interested in communication we briefly describe only Communication Layer.

Communication subsystem is divided into two parts. Fist of them is *mailbox*, which is responsible for receiving and dispatching incoming messages and also holds queue of outgoing messages. It consists of two threads, first, the reader thread, is holding FIFO incoming message queue with priority and second, the writer thread, works with outgoing queue of messages.

Writer thread periodically checks outgoing queue for messages to dispatch. For each found message it looks to the local address book for recipient. If recipient is found, writer serializes message into the sequence of ASCII characters and sends it via network socket. When no recipient address is found, the writer passes the message into holding buffer and asks known nameserver agents for the address. If address is found by some nameserver agent it returns the address back to the asking one and message is sent to the proper destination. When no address is found writer thread gets aware of that and passes error message into reader thread incoming queue.

Second part is *message handler*, which can be described as the agents internal sorting office. It continually checks incoming message queue and forwards messages to the relevant components of agent.

4.2 FIPA-OS

FIPA-OS is open agent platform from Nortel Networks. Within main concepts of this platform belongs openness, which means the ability to be interconnected with other FIPA compliant platforms. Openness is even emphasized with distribution under open-source licensing scheme. Other important feature of this platform is standards compliance. As the name suggest FIPA-OS is created under guidance of FIPA standards. These features makes it a suited tool for creating open, standards following agent systems and it has already been used in domains of virtual private network provisioning, distributed meeting scheduling and a virtual home environment (all mentioned in [18]).

FIPA-OS communication is based on ACL language and module responsible for processing messages is divided into four layers of components where are *conversation*, *ACL message*, *content* (syntax) and *ontology* (content semantics). This decomposition is done to support flexibility and needs of various agents because of heterogeneous world that puts various demands on their communication. FIPA-OS supports ASCII and XML encoding of ACL messages and there is also support for FIPA SL0 and FIPA SL1 content languages.

FIPA-OS support for conversation coordination is done via *conversation* layer. Its based on the assumption that single messages are most of the time meaningless and there is mostly need to handle more complex message exchange. Without dialog management messages are exchanged with no further context and it is more difficult to detect failures or inappropriate response occurrences. Firstly, conversation is defined as an instance of any FIPA interaction protocol. FIPA-OS uses defined field Conversation-ID to coordinate conversations. It is generated by sender agent who initiate conversation and it is composed from agent ID, time of message creation and counter, which ensures conversation uniqueness.

4.3 JADE

JADE is software framework that supports development of applications fully compliant with FIPA standards. Main purpose of JADE is to simplify agent system creation process through a comprehensive set of system services. To achieve this simplification JADE offers following list of features:

- FIPA compliant agent platform containing agent management system (AMS), directory facilitator (DF) and the agent communication channel (ACC). Note that DF can be started multiple times on different hosts to provide multi-domain environment.
- Distributed agent platform dividable on some hosts, which needs to run single Java Virtual Machine (JVM) each.
- Java API to send/receive ACL messages, which are represented as ordinary Java objects.
- FIPA97 compliant IIOP and HTTP protocols to connect different agent platforms.
- Library of FIPA interaction protocols ready to use.
- Graphical user interface to manage whole platform. Several tools are manageable from this gui for instance sniffer agent.

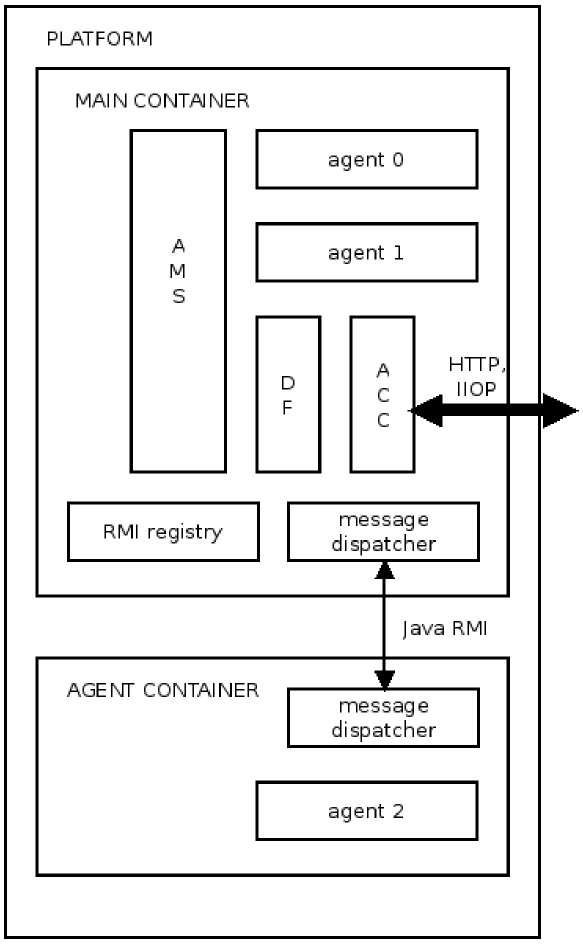

Fig. 2: Jade platform architecture

To understand JADE communication model, we need to know some basics about JADE software architecture depicted on Fig. 2. JADE platform consists of some containers (each provided by single JVM) where one is main among them. This main container contains AMS and ACC agents and is representing platform to the outside world.

Each agent container is a multi-threaded environment providing one thread for each agent plus some system threads created for dispatching messages. Each of these containers is created as RMI (remote method invocation) object that controls life cycle of its agents and their creation, state and killing. Besides that containers also manage dispatching incoming messages and putting them into proper agents message queue. To the other side, when sending outgoing messages, containers look up receiver agent location and pick suitable type of transport for ACL message delivery.

When the platform (main container) starts, it creates an internal RMI registry listening on specified TCP/IP port. RMI registry is basically a table containing RMI object reference and when a common container starts it looks up for specified host to join and if found it registers itself with main container RMI registry and thereby joins agent platform. Besides that main container holds Agent Global Descriptor Table where each agent name, AMS data and its container's RMI reference are stored. It is worth mention that each container holds cache of contacted agents on other containers. This is done for performance reasons.

When sending a message, three scenarios are possible.

1. In first scenario agents exchanging messages live on the same container. In this case Java events are used to send a message. Java *ACL Message* object, which represents current message is simply cloned and given to recipient agent.
2. This scenario assumes that two agents live within the same platform but in different containers. In this case Remote Method Invocation is used to send a message. This avoids marshaling and unmarshaling Java objects and represents clear way to pass object possibly through network.
3. In the last scenario messages are exchanged between two platforms. In this case is necessary to use one of the protocols supported by JADE for inter platform communication. There belongs HTTP protocol and IIOP protocol with OMG IDL interface. By use of IIOP message is translated from Java object to Java string and consequently to byte stream. On the other side opposite procedure takes place. If the second platform is not JADE then it must understand IIOP protocol and translate it to something it understands. If HTTP protocol is in use than XML is used for encoding the message.

The platform represents single interface to the outside world by the use of FIPA ACC standard agent. This agent is in fact the CORBA/IIOP server or HTTP server, which depends of Message Transport Protocol (MTP) used. When it receives ACL message encoded as a string (possibly from non-JADE agent) it converts it to *ACLMessage* object and sends it to the proper agent inside the platform and vice versa.

5 Towards FIPA standards

This chapter describes possibilities of interconnection PN agent and JADE. As said before we are dealing with two different programming languages and their cooperation with each other. To beat this challenge we need to use network communication, which does not rely on concrete programming language. Other restriction is that both languages need to actually support chosen technology.

Two scenarios appear to be usable. In first we create a proxy agents in JADE and in PN agent. They will be used as a bridge for messages from PN agent to JADE agents and vice versa. In second scenario we connect two different platforms by use of HTTP as a MTS protocol.

5.1 Proxy agent

First we can disclose that at the end we decided to implement the entire functionality in squeak so solution with proxy agent represents just a possibility. Nevertheless solution with proxy agent is more generic than the other because it doesn't depend on any technology used. It presents the concept of ordinary agent within MAS, which is given additional functionality to communicate via newly programmed channel. This approach has only one limitation. It assumes that programming languages, platforms are written in, has at least one network protocol that they both can handle.

When connecting JADE with PN agent we are able to use network sockets for purposes of connection as in Fig. 3. Here we evade using of standard Agent Communication Channel and we establish connection of our own. Problem is that via network sockets we can establish only connection, not communication. To establish communication we need common agent communication language. Since JADE and PN agent both use ACL communication language we have no problem. More precisely PN agent operates with ACL in the semantics scope so there was necessity to add syntax parser but this problem we needed to solve anyway.

The problem will occur if two platforms were using different language each. In this case there will be need to develop a translator from one language to the other and vice versa. This translator can be programmed as the insides in one of proxy agents. In this scenario messages are sent in textual form and on one side proxy agent translates it between languages. Next problem is that different languages can have various expressive power. In this case there is necessity to add support for transferring additional informations that can't be expressed by translation. Question is if an additional information is any use in target platform. Since target language does not support it some adjustments are necessary in target platform. Here we are getting to the point, where ends programming agents on user level and we are getting to the point where adjustments in target platform are necessary. This means that we need access to target platform source code and possibility to change it. Other way how to evade various expressive powers of languages is ignoring additional information that language with more power gives.

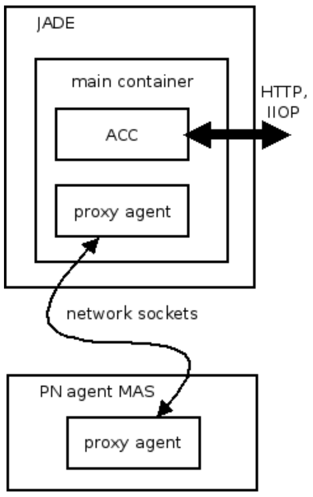

Fig. 3: Communication via proxy agent

5.2 Interconnecting on platform level

Fist step in platform level interconnection is to start two platforms and establish connection between them. For purposes of inter platform communication JADE offers two protocols. First is IIOP/CORBA protocol and second is HTTP protocol. Since we are creating solution as flexible as possible we wanted to use IIOP protocol because is marked by FIPA as mandatory. Problem is that to this day squeak lacks in IIOP support. That is why we were forced to use HTTP protocol, which support in squeak is unarguable. Connection via HTTP is shown on Fig. 4 and presents solution, which brings partial FIPA compliance to PN agent platform by supporting inter platform communication with use of FIPA ACL message exchange.
Positive information is that FIPA defines inter platform communication in textual form. That is useful feature we can exploit even by using different programming language on each side of connection because we don't need to care about marshaling and unmarshaling objects sent, which is most of the time language specific process. Since FIPA defines textual form of the message sent we can show message insides by packet sniffing. This way is easily explainable what are necessary parts when composing message for any JADE agent.

HTTP is client/server request-response protocol and by use of JADE and PN agent both sides of communication need to manage HTTP server and also HTTP client.

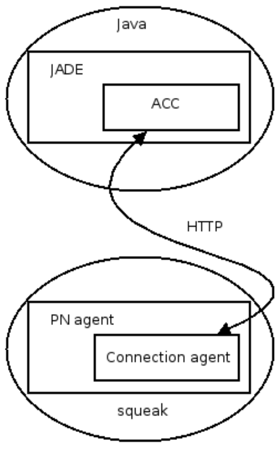

Fig. 4: Communication via HTTP

In HTTP request there are data, which client wants to give to server, therefore POST method is used. In the following header we can see, what information is specified by uploading message to the server side.

```
POST http://sycho-laptop:7778/acc HTTP/1.1
Cache-Control: no-cache
Mime-Version: 1.0
Host: sycho-laptop:7778
Content-Type: multipart/mixed ;
boundary="643caab056c1109c6f0d895cd67e5a4"
Content-Length: 1094
Connection: Keep-Alive
```

First useful information we can see is that to create message for remote platform we need to know its network location. This address needs to be provided after the POST keyword and must be reachable within the network. Next important information is *Content-Type* field. As we can see JADE uses *multipart* type, which is abstraction of encapsulation more entities within a single message body. Subtype of multipart type is *mixed*. That allows putting various fields within a single message, which is useful for placing message envelope and message content into single request.
The message body consists of message envelope and message itself specified in ACL language. Message envelope is specified in XML and looks like following:

```
Content-Type: application/xml

<?xml version="1.0"?>
<envelope><params index="1">

  <to><agent-identifier>
    <name>pong@P1</name>
    <addresses>
      <url>http://sycho-laptop:7778/acc</url>
    </addresses>
  </agent-identifier></to>


  <from><agent-identifier>
    <name>ping@P2</name>
    <addresses>
    <url>http://izakjakub-desktop:7778/acc
</url>
    </addresses>
  </agent-identifier></from>

  <acl-representation>
    fipa.acl.rep.string.std
  </acl-representation>
  <payload-length>236</payload-length>
  <date>20100603Z003837180</date>

  <intended-receiver><agent-identifier>
    <name>pong@P1</name>
    <addresses>
    <url>http://sycho- laptop:7778/acc</url>
    </addresses>
  </agent-identifier></intended-receiver>

</params></envelope>
```

As you can see envelope is divided into several parts. Most important are first two parts, which specify Message sender and message receiver. Information about agent name is provided via FIPA defined *name@platform* format and URL specified consequently is address of opposite agent platform. Envelope continues with some additional information about class, which was used to encode message, then is size of payload for control purposes and at last is date. Last logical unit is *intended receiver* parameter. This parameter specifies agent that will receive a copy of message. If it is not specified then JADE creates intended receiver (receivers) from agent (or list of agents) defined in `<to>` parameter.
Last part of message is ACL message itself. For purposes of presentation was chosen simple message that contains as few information as possible. Since message is encoded to ACL language it strictly follows its syntax thus presenting rich message would only bring useless complexity. ACL part of the message content looks like follows:

```
Content-Type: application/text

(INFORM
 :sender  ( agent-identifier :name
ping@P2  :addresses (sequence
http://izakjakub-desktop:7778/acc ))
 :receiver  (set ( agent-identifier :name
pong@P1  :addresses (sequence http://sycho-
laptop:7778/acc )) )
 :content  "ping"
)
```

In message we can notice four useful information. First is message performative, which is in this case *INFORM* performative. Next are expected fields such as *sender*, *receiver* and *content*.

By all this knowledge gathered we can easily simulate creation of such message and by that accomplish message exchange between JADE and PN agent platform.

6 Conclusion

In this paper we presented our research in the area of agent platform interconnection. We briefly outlined some of the FIPA principles and standards necessary to understand what means FIPA compliance. Since we are interested in interconnecting platforms we described some of the FIPA compliant platforms and their messaging subsystems. It turns out that if not all of them then most of the FIPA compliant platforms are written in Java programming language. This fact makes their interconnection easier than we faced.

We have tested our solution with simple ping pong agents that exchange a message and at the end we can claim that throughout HTTP protocol there is possibility to interconnect JADE with PN agent platform, which is not written in Java but in smalltalk, concretely in its squeak implementation.

It is worth to mention that we are at the beginning of a long journey to be FIPA compliant but even this added ability to exchange messages widens PN agent use.

7 References

[1] Charlton P., Cattoni R., Potrich A., Mamdani E. *Evaluating the FIPA standards and its role in achieving cooperation in multi-agent systems*, “Multi-agent systems, Internet and applications”, HICS-33 software technology minitrack, Mawi, Hawaii, January, 2000.

[2] Bellifemine F., Poggi A., Rimassi G., *JADE: A FIPA-Compliant agent framework*, In: Proc. Practical Applications of Intelligent Agents and Multi-Agents, pages 97-108, April, 1999.

[3] Patil R.S., Fikes R.E., Patel-Scheneider P.F., McKay D., Finin T., Gruber T., Neches R. *The DARPA knowledge sharing effort: progress report*. In: Proc. Third Conf. on Principles of Knowledge Representation and Reasoning, Cambridge, MA., pages 103-114, 1992.

[4] *Object Management Group* [online], [cit. 2010-06-10], <http://www.omg.org >.

[5] *Welcome to the Foundation for Intelligent Physical Agents* [online], [cit. 2010-06-10], <http://www.fipa.org>.

[6] Frost H. R., Cutkosky M. R., *Design for Manufacturability via Agent Interaction*, Paper No. 96-DETC/DFM-1302, In: Proceedings of the 1996 ASME Computers in Engineering Conference, Irvine, CA, pages 1-8, August 18-22, 1996.

[7] Searle J. R., *Speech Acts*, Cambridge University Press, Cambridge, 1969.

[8] Finin T., Labrou Y., *KQML as an agent communication language*, J.M. In: Bradshaw

(ed.), Software Agents, pages. 291-316, Cambridge, MA, 1997.

[9] *FIPA Contract Net Interaction Protocol Specification* [online] [cit. 2010-06-10], <http://www.fipa.org/specs/fipa00029/SC00029H.html>.

[10] Mazal Z., Kočí R., Janoušek V., Zbořil F., *PNagent: a Framework for Modelling BDI Agents using Object Oriented Petri Nets*, In: Proceedings of the 8th ISDA. IEEE Computer Society.

[11] Bellifemine F., Caire G., Greenwood D., *Developing Multi-Agent Systems with JADE*, Chichester, West Sussex: John Willey & Sons Ltd., 2007, pages 286, ISBN 978-0-470-05747-6.

[12] Cowan D., Griss M., *Making Software Agent Technology available to Enterprise Applications* [online] [cit. 2010-06-10], <martin.griss.com/pubs/agents-j2ee.pdf>.

[13] Greenwood D., Lyell M., Mallya A., Suguri H., *The IEEE FIPA Approach to Integrating Software Agents and Web Services*, In: AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, Article No.: 274, 2007

[14] Horáček J., *Platforma pro mobilní agenty v bezdrátových senzorových sítích*, diplomová práce, Brno, FIT VUT v Brně, 2009.

[15] Zbořil F., *Low Level Language for Agent Behaviour Control*, In Proceedings of XXVIIth International Autumn Colloquium ASIS, pages 138-143, Ostrava CZ, 2005, ISBN 80-86840-16-6.

[16] *Publicly Available Implementations of FIPA Specifications* [online] [cit. 2010-06-10], <http://www.fipa.org/resources/livesystems.html>

[17] Nwana H. S., Ndumu D. T., Lee L. C., Collis J. C., *ZEUS: A Toolkit for Building Distributed Multi-Agent Systems*, Applied Artifical Intelligence Journal, volume 13, pages 129-186, 1999.

[18] Poslad S., Buckle P., Hadingham R., The FIPA-OS agent platform: Open Source for Open *Standards*, 2000

[19] *Minimal FIPA and FIPA Compliance Levels Work Plan* [online] [cit. 2010-06-10], <http://www.fipa.org/docs/wps/f-wp-00018/f-wp-00018.html>.