Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 38(3), 2011, Pages 46–55
ISSN: 1223-6934, Online 2246-9958

Collaborative software agents based on JADE for distributed data processing

Claudiu Ionut Popirlan

Abstract. The collaboration process among individuals with heterogeneous skills in a distributed virtual environment represents a crucial element of the extended enterprise. In last year contact centers become more complex in their function and organization so that the data processing becomes more formal to ensure consistency and efficiency. This research proposes a preliminary look at data processing in a contact center with complex architectures, by introducing collaborative software agents. A multi-agent system is employed to coordinate tasks and information stored in heterogeneous resources. The system architecture is first discussed in this paper. The JADE (Java Agent DEvelopment Framework) implementation of the system provides an environment to coordinate data processing and integrate rules, optimization and simulation models.

2010 Mathematics Subject Classification. Primary 68T42; Secondary 68T35.
Key words and phrases. Software agents, collaborative agents, multi-agent system, JADE, data processing.

1. Introduction

Today every enterprise uses electronic information processing systems (production and distribution planning, stock and supply management, customer and personnel management) and usually these systems are coupled with a database system (e.g. databases of customers, suppliers, parts etc.). However data processing alone is not enough. General patterns, structures, regularities go undetected and often such patterns can be exploited to increase turnover. The evolution of communication services over the past century has spawned a broad new industry known as electronic contact, which provides electronic communication mechanisms between people and businesses or organizations. A contact center (also referred to as a customer interaction center or e-contact center) is a central point in an enterprise from which all customer contacts are managed.

Contact centers are the contemporary successors of call centers [11]. The contact center typically includes one or more online call centers [6], but may include other types of customer contact as well, including e-mail newsletters, postal mail catalogs, web site inquiries and chats and the collection of information from customers during in-store purchasing. A contact center is generally part of an enterprise's overall customer relationship management (CRM). The modern contact center is a complex socio-technical system. Some view contact centers as the business frontiers but others as the sweat-shops of the 21st century.

The design of the modern contact center and the management of its performance surely must be based on strong scientific principles [10]. This is manifested by a growing body of academic multi-disciplinary research, devoted to call centers, and ranging

Received June 17, 2011. Revision received August 18, 2011.

from Mathematics and Statistics, to Operations Research, Industrial Engineering, Information Technology and Human Resource Management. In the context of contact center, multiple users with concurrent access to multiple system resources have to collaborate in a distributed design environment in order to achieve global optima.

Software agents originally were discussed in the 70's and in the mid 90's briefly gained some momentum but then stalled. The "software agent" term has found its way into a number of technologies and has been widely used, for example, in artificial intelligence, databases, operating systems and computer networks literature. All definitions agree [15, 7, 19] that an agent is essentially a special software component that has autonomy that provides an interoperable interface to an arbitrary system and/or behaves like a human agent, working for some clients in pursuit of its own agenda.

Most discussions on agents focus on their autonomy, intelligence, mobility and interaction [16, 17, 12, 13, 14]. Agent-based systems [1] claim to be next generation software capable of adapting dynamically to changing business environment and of solving a wide range of knowledge processing application. Although sophisticated software agents can be difficult to build from scratch due to the skills and knowledge needed, the widely available agent construction toolkits may provide a quick and easy start to building software agents without much agent expertise. For example, JADE (Java Agent DEvelopment Framework) [1, 8] is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications [4] and through a set of graphical tools that supports the debugging and deployment phases. Significant research and development into multi-agent systems has been conducted in recent years and there are many architectures available today. Since 1997 a few papers have been written, and even a commercial framework or two have emerged. Nevertheless, several issues still need to be faced to make the multi-agent technology widely accepted:

- secure and efficient execution supports;
- standardization;
- appropriate programming languages and coordination models.

In this paper is presented a multi-agent approach using our preview results [16, 17, 12] for distributed data processing, in context of contact centers. We introduce a collaborative software agent's model for data process in contact center data bases infrastructures. Due to the use of common development and agent communication methodologies, collaboration has been formed to take advantage of potential overlapping objectives. The goal of the collaboration is to further develop the demonstration capabilities so that agents are able to exchange messages and share knowledge with each other. Accordingly, the paper presents the design, architecture and implementation of the multi-agent system using JADE. An adequate example shows the effectiveness of the Java-based software agents approach.

2. Collaborative Software Agents in JADE

2.1. JADE - Medium Description. Multi-agent systems can be realized by using any kind of programming language. In particular, object-oriented languages are considered a suitable means because the concept of agent is similar with the concept of object. In fact, agents share many properties with objects such as: inheritance, message passing, encapsulation, etc.

Agent-oriented programming languages are a new class of programming languages that focus on taking into account the main characteristics of multi-agent systems. Minimally, an agent-oriented programming language must include some structure corresponding to an agent, but many also provide mechanisms for supporting additional attributes of agency such as beliefs, goals, plans, roles and norms. A list with several agent-oriented languages can be found in [12].

Software platforms and frameworks are the other key means enabling the development of multi-agent systems. Most provide a means to deploy multi-agent systems on different types of hardware and operating systems, usually providing a middleware to support their execution and essential operations such as communication and coordination. Some of these platforms and frameworks have the common goal of providing FIPA-compliant functionalities to support interoperation between different multi-agent systems.

JADE [1] is a software platform that provides basic middleware-layer functionalities which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction [19]. A significant merit of JADE is that it implements this abstraction over a well-known object-oriented language, Java, providing a simple and friendly API.

2.2. Collaborative Agents using JADE. One goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. During the development of the system with JADE, the following types of classes are created and implemented:

- Agent classes to describe various types of agents.
- User Interface classes for customer interaction.
- Agent activity classes for behaviors.
- Database classes to handle the database of the system.
- Communication classes to manage the negotiation between agents.
- Ontology classes to define concepts, predicates and agent actions for the domain.

To conclude, the basic software agent structure is shown in Figure 1.

The system must be able to make possible the agents communication, including the communication protocols. Agent's communication is based on:

(1) Sending (action);
(2) Receiving (perception) messages.

The degree of coherence and coordination comes from the extent to which the system avoids redundant actions, competition on resources, bottlenecks and the unsafe operating conditions. The goal is to maintain an overall coherence, without having always a global control in place. The coordination between mobile agents not entering the competition is based on cooperation. For the agents entering in the competition, or those having reciprocal dependence, the coordination is based on negotiation.

Communication between agents is done via messages. An agent can communicate with other agent by message passing. An agent that wants to communicate with another agent first has to create a message object and then send it to the target agent. A message object has a kind and an optional argument object. The receiver agent determines what to do by checking the kind of received message and get parameters as the argument object. For system implementation one can use a subset of standard indicators [5] of the Knowledge Query and Manipulation Language (KQML):

```
tell
:content <expression>
:language <word>
```

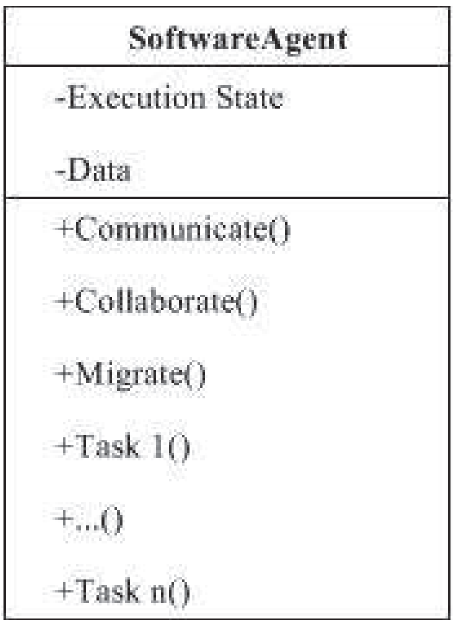

FIGURE 1. Software Agent Basic Architecture

```
:ontology <word>
:in-reply-to <expression>
:force <word>
:sender <word>
:receiver <word>
```

A message has two sections:
- message header,
- message content (also known as message body).

The header contains the information regarding:
- Sender,
- Receiver(s),
- Subject,
- Date,
- Time,
- Priority.

There is also a slot that contains the agent identification. Using the agent identification, the information about an agent, such as its name, can be retrieved from a repository or agent directory. Message content has the following attributes:
- an action verb,
- an object,
- preconditions,
- constraints.

The action verb is used to indicate the type of action to be taken by the receiver, such as
- request,

- propose,
- query.

An object is the result or expectation. Agents must have intelligence to process messages. A body of knowledge supports the intelligence of an agent.

2.3. Contact Center - Typical Process. A contact center would typically be provided with special software that would allow contact information to be routed to appropriate people, contacts to be tracked, and data to be gathered. The contact center architecture is presented in many studies, for example [11, 6, 12]. A typical process in a contact center can be resumed as follow: The customer dials the call-center number and is greeted with a number of options that include the following:

- a recorded message followed by the placement in a telephone queue managed by an Automated Call Distribution System (ACD);
- an Integrated Voice Response (IVR) that offers the caller different options where caller interacts with the IVR using a touch-tone telephone or voice control;
- the call is immediately directed by an ACD to an agent who manages the query.

If the agent cannot personally resolve the query they direct the call to someone who can answer the query. In this typical process we will be able to search the data from a distributed data base system. In this respect, the collaborative software agents visit, one after another, all or a part of the data base servers to whom they ask for certain information.

3. Using Agents in Distributed Data Processing

In JADE multi-agent systems are built using a variety of techniques. For our architecture example, we use a simple requirements, analysis, design, and development flow, as shown in Figure 2.

The agents include a design agent, a group of process planning agents, a repository agent, and a execution control agent. Humans interact with the agents through graphical user interfaces (GUI). An execution agent is a kind of information agent that has one particular role, which is to find one or more items in a data base. It can be formally implemented as a goal-based or utility-based agent. For our architecture, we use the Roles Model method of analysis, which is a model within the Gaia Agent Design Methodology [20]. It describes what a particular agent does, why it does it, and what responsibilities and permissions it has:

- To search the data (attribute value for a certain object) in contact center with distributed data bases;
- Read and search data from any open data base source;
- Responsibilities: liveness, safety.

The input from the user (customer query) is in simple plain text, and our multi-agent systems, described in [16] usually talk in local database. The "*BuildQuery*" activity takes the data and generates a simple query based upon the local data found in the previous activity. "*ExecuteQuery*" then executes the query and determines if the result is useful. If it is, the "*InformUserOfAnswer*" shows the result to the user. If not, the simple agent starts again with different data base (in distributed context, other location). The sub-goals described in Figure 1 will be translated into simple public methods in Java, whereas the overall goal is translated into a JADE Behaviour. We use an object-oriented class diagram to transcribe the goal flow diagram into a form that is ready for development in an object-oriented language such as Java, as shown in Figure 3. It refers to Agent and "*OneShotBehaviour*", which are parts of the

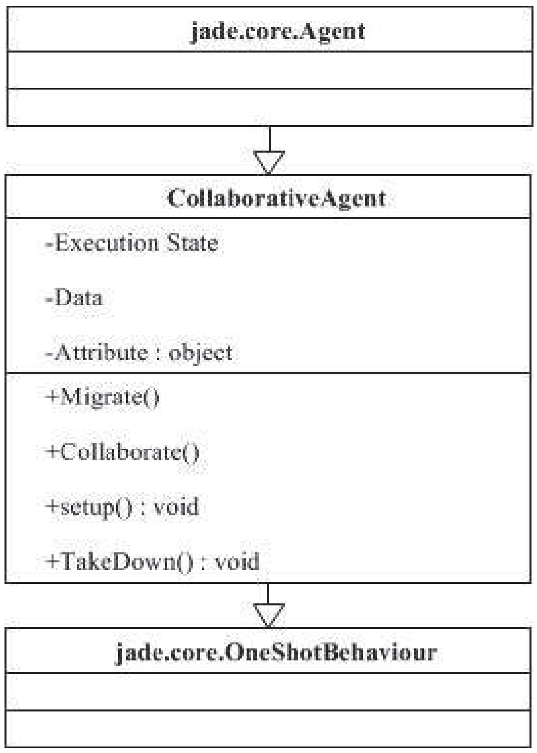

FIGURE 2. Software Agent Basic Architecture

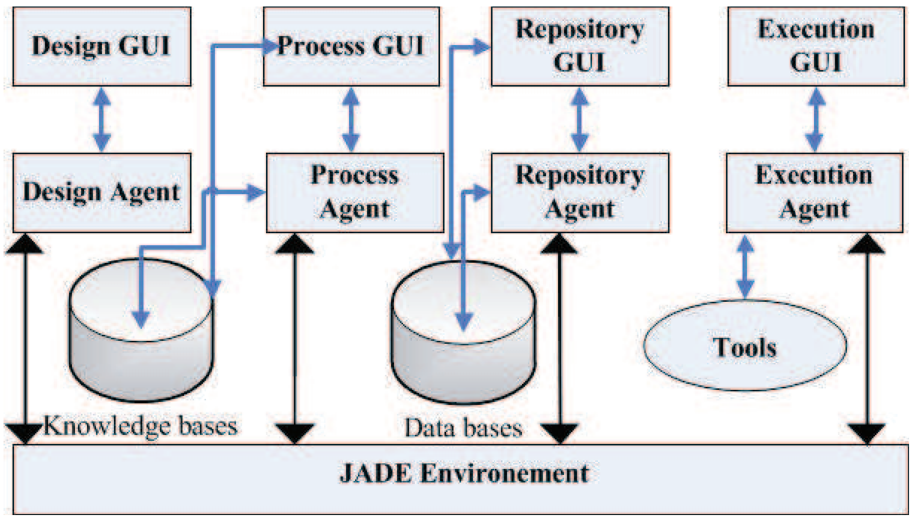

FIGURE 3. Object-oriented class diagram for data processing

JADE library. The main class within the program is the "*CollaborativeAgent*", which is of type Agent in the JADE library. An agent lives in a particular environment, and therefore JADE provides one. JADE also provides a graphical user interface (GUI) to initialize, control, and terminate agents, as shown in Figure 3.

4. Case Study and Simulation Results

In order to illustrate the proposed solution for searching data in contact center with distributed data bases, we will explain the setting and describe a case study. The application exploits collaborative agents to reach remote distributed data bases, in contact center context, and locally access data of interest, analyzing them and extracting the required information without any need to transfer the data over the network. For instance, an agent sent to a remote data base can analyze the local data and come back with the attributes that contain a specific keyword. To speed up the research, the application can be shaped after a tree of collaborative agents. An agent lives in a particular environment, and therefore JADE provides one. Shown below is a sample of the source code included in this application, created in JADE and written in Java:

```
//...
public class CollaborativeAgent extends Agent{
public JTextArea message;
//...
class Knowledge extends TickerBehaviour{
//...
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
AMSAgentDescription [] agents = null;
try {
Collaborate();
}
catch (Exception e){
System.out.println("Exception 1: "+e);
}
//...
try {
Communicate();
} catch (IOException ex) {
Logger.getLogger(KnowledgeType.class.getName()));
}
//...
}
//...
}
```

If an agent on a local data base finds links to other possibly interesting data on different location (distributed context), it clones itself and has the clones follow these links, to recursively continue the search work on different data bases, as shown in Figure 4.

The main indicator for efficiency, in contact center, is the productivity [9], measured over a certain period (for example, a week). It is usually given, as in (1), as the percentage of time that an agent is working of his or her total scheduled working time.

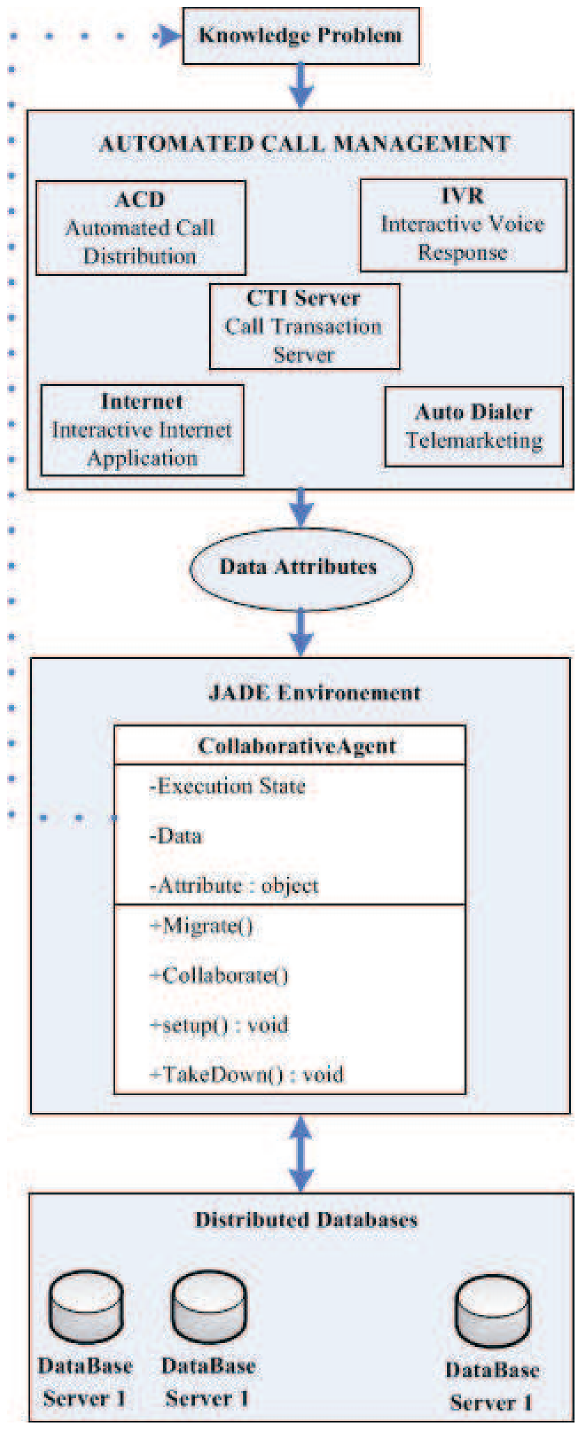

FIGURE 4. Using collaborative agent in data processing

$$Productivity = \frac{TWT}{TWT + TA} \cdot 100\% \tag{1}$$

where TWT is total working time and TA is time available. The percentage of calls that is answered in less than a certain fixed waiting time is sometimes called the telephone service factor (TSF). Another commonly used waiting time metric is the average speed of answer (ASA). The Erlang C formula [9] gives the TSF, and can be used to compute the average waiting time for a given number of human agents (operators), service times and traffic intensity. In order to prove the efficiency of solution proposed we will take into account the same scenario that was presented in [16] with same values. The "*CollaborativeAgent*" creates only one clone ("*CloneAgent*") for each of the Data Base Servers (3 clone). The data processing duration of each clone is presented in Table 1.

5. Conclusion and Future Work

The growth of contact centers, e-commerce, and more complex value chains has raised additional issues of enterprise data management and exploitation, while demonstrating beyond doubt that available data are insufficient to support new customer management processes. In this paper a multi-agent approach for distributed data searching in context of contact centers was presented. We provide a description of collaborative software agents, developed in JADE environment, and we presented a simple illustration to show how the proposed system might work.

As a further, we will try to implement agents in order to adopt a control-oriented point of view, for example to request services while they require files. With the adoption of a blackboard or a tuple space on each data base server, data can be accessed without requiring the presence of peculiar services and in a more natural data-oriented style. We intend to develop a prototype of this multi-agent system, which can demonstrate that more practical and relevant problems can be addressed successfully.

Acknowledgment

This work was supported by the strategic grant POSDRU/89/1.5/S/61968, Project ID61968 (2009), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007-2013.

References

[1] F.L. Bellifemine, G. Caire and D. Greenwood, *Developing Multi-Agent Systems with JADE*, Wiley, 2007.

[2] R.H. Bordini, L. Braubach, M. Dastani, A.E.F. Seghrouchni, J.J. Gomez-Sanz, J. Leite, G. O'Hare, A. Pokahr and A. Ricci, A Survey of Programming Languages and Platforms for Multi-agent Systems, *Informatica* **30** (2006), no. 1, 33–44.

[3] Eurepean Co-ordination Action for Agent Based Computing, website: `www.agentlink.org`

[4] FIPA, Foundation for Intelligent Physical Agents, website: `http://www.fipa.org`

[5] T. Finin, Y. Labrou and J. Mayfield, *Software Agents: KQML as an agent communication language*, invited chapter in Jeff Bradshaw (Ed.), MIT Press, Cambridge, 1997.

[6] N. Gans, G. Koole and A. Mandelbaum, *Telephone call centers: tutorial, review and research prospects*, Manufacturing and service operations management 5, 2003, 79–141.

[7] M.R. Genesereth and S.P. Ketchpel, Software Agents, *Communications of the ACM* **37** (1994), no. 7, 48–53.

[8] JADE, Java Agent Development Environment, website: `http://jade.tilab.com`

[9] N.M. Karnik and A.R. Tripathi, Design Issues in Mobile-Agent Programming Systems, *IEEE Concurrency* **6** (1998), no. 3, 52–61.

[10] G. Koole, *Call center mathematics*, online book `http://www.math.vu.nl/~koole/ccmath/book.pdf`

[11] A. Mandelbaum, A. Sakov and S. Zeltyn, *Empirical analysis of a call center*, Technical report, Faculty of Industrial Engineering and Management, Technion-Israel Institute of Technology, Haifa, Israel, 2001.

[12] C.I. Popirlan, Knowledge Processing in Contact Centers using a Multi-Agent Architecture, *Wseas Transactions on Computers* **9** (2010), no. 11, 1318–1327.

[13] C.I. Popirlan and L. Stefanescu, Mobile Agents System for Intelligent Data Analysis, *Proceedings of WSEAS Applied Computing Conference* (2009), 663–668.

[14] C.I. Popirlan and M. Dupac, An Optimal Path Algorithm for Autonomous Searching Robots, *Annals of University of Craiova, Mathematics and Computer Science Series* **36**(2009), no. 1, 37–48.

[15] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[16] G. Stoian and C.I. Popirlan, A proposal for an enhanced mobile agent architecture (EMA), *Annals of the University of Craiova, Mathematics and Computer Science Series* **37** (2010), no. 1, 71–79.

[17] N. Tandareanu and C.I. Popirlan, A Mobile Agents Approach for Knowledge Bases Processing, *Proceedings of the Twelfth IASTED International Conference on Intelligent Systems and Control (ISC 2009)*, Cambridge, Massachusetts, USA, November 2-4, 2009, 27–32.

[18] P. Thati, P.H. Chang and G. Agha, Crawlets: Agents for high performance web search engine, *Lecture Notes in Computer Science* (2001), 119–134.

[19] M.J. Wooldridge and N.R. Jennings, Intelligent Agents: Theory and Practice, *Knowledge Engineering Review* **10** (1995), no. 2, 115–152.

[20] M. Wooldridge, N.R. Jennings and D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, *Journal of Autonomous Agents and Multi-Agent Systems* **3** (2000), 285–312.

(Claudiu Ionut Popirlan) Department of Computer Science, University of Craiova, 13 A.I. Cuza Street, Craiova, 200585, Romania
E-mail address: `popirlan@inf.ucv.ro`