International Journal of Informatics and Communication Technology (IJ-ICT)
Vol.3, No.1, February 2014, pp. 1~12
ISSN: 2252-8776

Semantic Aware Multi-Agent System Advantages

Trajche Manev*, Sonja Filiposka*,**
* Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Skopje, Macedonia
** University of the Balearic Islands, Palma de Mallorca, Spain

Article Info

Article history:

Received Dec 12th, 2013
Revised Jan 20th, 2014
Accepted Feb 26th, 2014

Keyword:

Intelligent agents
Multi-agent systems
Software Agent
Semantic Web
FIPA
Jade
Schedule

ABSTRACT

Internet changes the way people communicate with each other and also affects the business flow. Technology changes affect how people live and communicate with each other, but they themselves also transform and become more human like. The agent oriented programming and multi-agent systems are relatively new example technologies that manifest these changes in a multifacet way. The goal of this paper is to explore the possibilities one can obtain by combining multi agent systems with the emerging semantic web technologies. This powerful combination leads towards new types of complex system development. Using a case study example, together with a comparison with the traditional system design approach, the paper also discusses the possibilities of semantic agent platforms and their development and shows how developers can benefit from following the standards to agent development.

Corresponding Author:

Trajche Manev
Faculty of Computer Science and Engineering,
Ss. Cyril and Methodius University,
Rugjer Boshkovikj no. 16, PO BOX 393, 1000 Skopje, Macedonia.
Email: trajche.manev@gmail.com

1. INTRODUCTION

The paradigm of agent oriented programming [1] produces applications defined as collection of components, so called agents. When an agent is defined, there are some properties that it should exhibit. The first and foremost property is autonomy, which means operating without any direct intervention of humans. Second is social ability, describing the ability to interact with other agents and/or humans. Third is reactivity, which involves perceiving the environment and responding to any changes that occur within. Finally pro-activeness, this meaning that the agent is supposed to have a goal-directed behavior.

As described in [2], the term “agent” or software agent is present in many technologies and is widely used (i.e. in artificial intelligence, operating systems, computer networks, etc). In computer science, a software agent is a program, which acts on behalf of a user, or some other program. Depending on the complexity of the problem, agents can work solo, or they can interact with other agents. By using multi-agent systems complex systems can be successfully modeled. When implemented as a comprehensive system, agents are capable of achieving highly sophisticated goals autonomously and, if written correctly, will continue to search for a solution until the goal is complete.

Semantic web technologies [3], on the other hand, offer a new way of managing information and knowledge. The possibilities for extending the ontologies, which represent the core for the semantic web, it allows developers to grow beyond the need of using an exactly specified rigid architecture for the solution. However, this also means that the Semantic Web heavily relies on formal ontologies to structure data for comprehensive and transportable machine understanding. Although the developed Web standards for expressing shared meaning have progressed over the years, fully autonomous agents can only flourish when standards are well established [4].

One multi-agent system is formed with the interaction of a number of intelligent agents in a given environment. Multi-agent systems can be used for solving problems, which are difficult or impossible for an individual agent. There are frameworks that help alleviate the development process [5].

Intelligent agents and semantic web services are two technologies with great potential. Striking new applications can be developed by using the tools and techniques they provide. However, semantic web services are needed for a higher software entity to be able to deal with them while, on the other hand, agent technology has historically suffered from a number of drawbacks that must be addressed. Integrating these two technologies in a joint environment can overcome their problems while strengthening their advantages. In this paper, the necessity for integrating these technologies and the potential benefits of their combination are analyzed.

In this paper we discuss the opportunities and possibilities offered by the sematically aware multi-agent systems by reviewing the existing modeling specifications and communication standards and, afterwards, designing and analyzing an example case study system in both the traditional approach and one involving agents and semantics. The goal of the paper is to stress the differences in these approaches and turn the attention towards the new possibilities offered by multi-agent systems that can increase the quality of experience of the end-user.

The rest of the paper is structured as follows: in the next section we introduce the FIPA specifications for multi-agent system design and talk about the importance of standardized communication between the agents. In Section 3 we add the semantic technology to the multi-agent systems, while in Section 4 we discuss the possibilities offered by the JADE framework as one of the options that offer development of semantic aware multi-agent systems. In Section 4 then we make a comparison of the semantic aware multi-agent system with a traditional one using an example case study we developed. In Section 5 we present our final conclusions.

2. AGENT MANAGEMENT

Some agent technologies have reached a considerable degree of maturity. In order to ensure effective and widespread use, agent technologies require standardization.

Standardization of generic technologies has been shown to be possible and to provide effective results by other standardization. The aforementioned sentences form the root facts of the Foundation for Intelligent Physical Agents (FIPA) specifications [6]. An important aspect of the agent systems, which refers to the initial FIPA specifications, is agent management [7]. It’s a framework where FIPA consenting agents can exist, operate and be operated. This makes the logical model for agent creation, registration, locating, and communication between agents. Figure 1represents the agent management model and its components.

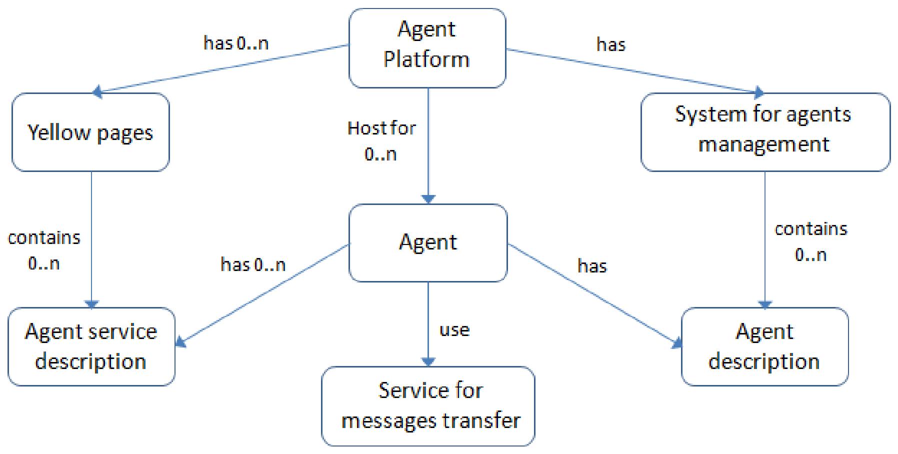

Figure 1. Agent’s management model

Agent Platform: Represents the physical infrastructure where agents are found. Platform consists of machines, operating systems, FIPA agent components, agents themselves and additional software for support. The inner design of the agent platform is left to be created by the developers of the agent system and is not subject to the FIPA standardization. One agent platform can be present on many computers, but agent residents do not have to be located on the same platform.

Agent: Agent is a process that populates the agent platform and usually gives one or more services, which can be published with proper service description. The design of these services is not meant to be designed by FIPA as long as it complies with FIPA specifications. FIPA only imposes the structure and the encoding of the messages used for information exchange between the agents and some other technologies. The agent must have one owner and must support at least one identity, which can be described by the use of the FIPA agent identifier (AID). It is used to name the agent, so it can be unambiguously distinguished. The agent can be registered on a wide range of transport addresses, from where it can be contacted.

Yellow pages - Directory Facilitator (DF): DF is an optional component on the agent platform. Its role is to offer service which is similar to the yellow pages and this service is available to the agents. It holds accurate information, complete time list of the agents and gives the current information about the agents in their own directory on unambiguously basis to all authorized agents. The agent platform can support any number of DF, which can be connected with another platform, and thus form federations of agents.

Every agent who wants to publish its services to the other agents must find the proper DF and ask for registration of its description. After the act of registration, the agent does not have any future obligations. Later, the agent can decide to ask for an end of the registration of the service description in any time. Additionally, the agent can make a request for searching through DF to find some descriptions by a given searching criteria. The DF can limit the access to the information in their own directory and can check for access by mapping all the permissions for access from the agent to the conditions of the other agents.

Agent management system (AMS): This is the main component of the agent platform and is responsible for managing the operations in the agent platform like creation and deletion of agents, migrations of agents from one to another platform. Every agent must register in the agent management system, so it can be provided with an AID. The directory holds all information about all of the agents from the platform and their current condition (active, suspended or waiting). Agent descriptions can later be modified by authorization of AMS. The life of a given agent on an agent platform ends with its request for end of the registration. After the end of the registration, the AID for the agent can be removed from the directory and will be available to the other agents who would like to register.

The agent management system can ask from the agent to perform a specific management function like, for an example, end of execution, and has the power to force the execution of a given operation in the case when the request is ignored. There can be only one management system by agent platform and if the agent platform covers several machines, the AMS is the authority for all of those machines.

Message Transport Service (MTS): This service of message transport is allowed by the agent platform so it can make transfer of FIPA-ACL messages between the agents on any agent platform and between agents on different agent platforms. Messages provide transport envelope, which has a few parameters.

3. THE SEMANTIC WEB KNOWLEDGE MANAGEMENT

Knowledge management deals with acquiring, accessment and support of the knowledge in one organization. In the big business it is key activity, because the internal knowledge is seen as intellectual asset, that can be used in order to obtain higher productivity, create new value and increase the organization competitiveness. However, most of the information is available in poorly structured form (e.g. text, audio, and video). From the knowledge management point of view the old web technology has a number of limitations [8] that include:

1. Information searching - companies usually depend on key based search,
2. Extracting information – lots of time and effort are needed in order to look for the information in the founded document. At the moment, intelligent agents are not able to do this kind of tasks,Information maintenance - at present there are problems like inconsistency in terminology and errors in removing old information,
3. Detecting information - new knowledge exists in corporate databases, and it can be drawn via data mining. This task is still difficult for distributed collections of documents.

The purpose of the Semantic Web is to provide the means for creating a more advanced system for knowledge management consisting of:

1. Concepts – the knowledge will be organized in a conceptual manner according to the meaning,
2. Automation - automated tools that allow for maintenance while checking for consistency, but that also extract new knowledge at the same time,
3. Questioning - key based searches can be replaced with answers to a human like given question. The requested answers (knowledge) will be pooled and presented in a human friendly way,
4. Wide search - the presented answer will be drawn and supported by sources that come from a given number of different documents,
5. Limited access – offering possibility to define who can see certain parts of information in the documents.

Decentralization and openness are inherent properties of multi-agent systems. The technologies they provide are thus the right abstraction for developing Web-oriented applications. Moreover, different works ([9], [10]) have proposed to use Semantic Web technologies for representing various dimensions of multi-agent systems (e.g., interaction protocols, norms, organizations). As described in [21] to be able to express the product design knowledge, first need to be analysis of the design knowledge source.

Given these facts, we believe that it is time to continue forward with the integration of the semantic web technologies and the multi-agent systems in order to improve reusability of data, knowledge, coordination strategies, etc. on the Web and across systems.

4. JADE PLATFORM FOR DEVELOPING AGENTS WHO CAN UNDERSTAND FACTS AND KNOWLEDGE FOR A GIVEN ONTOLOGY

4.1. FIPA Importance for JADE

FIPA is concerned with the standardization of the external behaviours of the system components, which need to be specified, leaving the internal architecture and the details for implementation to the developers. This kind of arrangement facilitates the cooperation between different platforms that can be used for implementation. In this paper we review the JADE platform [9], which complies with the FIPA2000 specifications (communication, management and architecture), thus providing a stable framework where FIPA agents can exist, operate and communicate.

4.2. Ontologies and content languages

Information, which is transferred and exchanged between the agents, is represented in a format of an ACL message that contains a special field for the message content. According to the FIPA specifications the content of the message can be a string or an array of bytes. In simple scenarios it can consist of only one value (e.g. phone price), but in real scenarios agents have the need for much complex communication. In the cases when more complex information is presented (e.g. for the case of selling mobile phones the information that the agents will need to exchange will consist of phone brand, model, price, on sale, etc.) there is a necesity for a well-defined syntax in order to ensure that the message content can be successfully translated and processed on the receiving side.

The FIPA syntax is also known as content language. FIPA does not force the developer to use any specific content language, but recommends the use of semantic language for communication. Example of semantic information for the action of selling in the previous phone example can be given by: Sell(:phoneModel “HTC Sensation” :phoneBrand “HTC” :price “200€”). When this message arrives at the agent, syntax parsing needs to take place in order for the agent to understand the meaning of the information. Thus, there must be some mutual understanding between the sender and the receiver. The set of attributes and symbols (Sell, phoneModel, phoneBrand, price) used for expressing meaning in the sentence is known as ontology. Ontologies are specific for a given domain (i.e. the example sentence has no meaning when selling cars) and need to be shared between the communicating agents.

Every time when a message is exchanged, the following actions needs to transpire: (1) The sender always needs to convert his internal representations into a valid ACL content expression and the receiver will have to do the opposite. (2) The receiver needs to perform a semantic validation in order to determine whether the received information conforms to the devised rules, which are described in the ontology shared between the agents.

JADE handles these conversions and checks automatically as is represented in Figure 2.

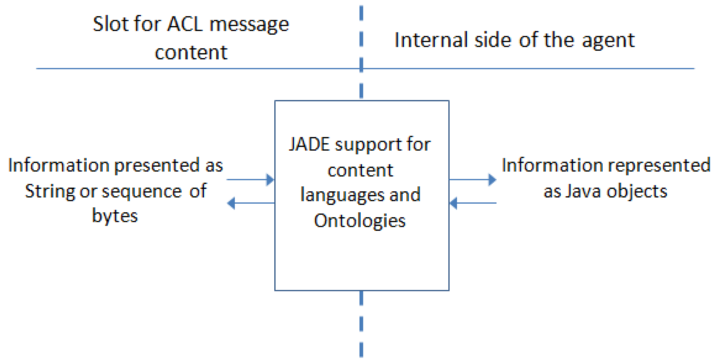
Slot for ACL message content
Internal side of the agent
Information presented as String or sequence of bytes
JADE support for content languages and Ontologies
Information represented as Java objects

Figure 2. JADE support for content languages and Ontologies

The conversion of the content into Java object enables easier manipulation with information without any additional transformations on the received data [11].

4.3. Defining Ontologies

The ontology in JADE is an instance of the jade.content.onto.Ontology class where the basic types of predicates, agent actions and concepts, which apply to the specific domain, are defined [11]. These schemes are instances of the classes PredicateSchema, AgentActionSchema and ConceptSchema, which are included in the packet jade.content.schema. In each one of these classes there are methods where slots, structure for each predicate, agent action and the concept can be defined.

Ontology is a collection of schemes, which usually do not evolve during the lifecycle of the agent, so they can be declared as singleton objects and in this way they can be shared throughout the agents, which are residing on the same java virtual machine.

4.4. Combining Ontologies

JADE provides support for combining Ontologies [11]. It’s possible to define Ontologies, which extend other defined Ontologies, simply by setting a parameter in the constructor when the new ontology is created. It’s suggested that a dictionary pattern is to be used as given in Figure 3.

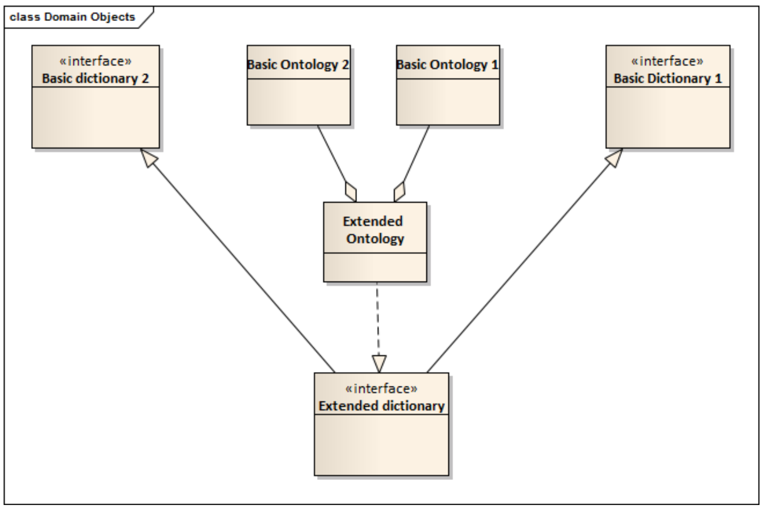
class Domain Objects
«interface» Basic dictionary 2
Basic Ontology 2
Basic Ontology 1
«interface» Basic Dictionary 1
Extended Ontology
«interface» Extended dictionary

Figure 3. Dictionary pattern

All the symbols used for names and concepts, predicates and agent actions are grouped into an interface, which present the dictionary. With this type of organization every constant from the dictionary can be accessed.

5. EXAMPLE CASE STUDY: MULTIPURPOSE SCHEDULING

The purpose of this case study is to show the developing approach involving agents using semantics and its benefits. As described in [20] the scheduling may become a difficult task: scheduling in hospitals, scheduling in transportation companies, scheduling of sport events and so on. Let’s assume we need a system for centralized appointments scheduling. For an example a given user wants to go on a massage, have a haircut, visit the dentist etc.

The user needs to make a call, or go to the appropriate web site in order to make an appointment. If the user does not find an adequate term at the desired location he needs to obtain information for other similar businesses and contact them sequentially. In order to make the user experience more transparent and bring him closer to the services provided, the intention of the system is to be a one centralized web application that will offer the user browsing and scheduling appointments in a number of different organizations that participate in the system. Thus the user will be viewing aggregated information from all organizations at once.

In order to pointout the benefits of using multi-agent semantic aware system, we will first discuss the implementation of this type of system in a traditional manner. Then we will compare the traditional implementation with the one done using semantic aware agents.

5.1. Traditional approach implementation

The developed system offers scheduling appointments for different types of services offered by a number of organizations working together in order to offer higher quality services to their customers. In order to implement the system in a traditional way, each organization needs to offer a web service that can be used for integration with a third party software like: Customer Relationship Management (CRM), e-commerce, users portal, etc.

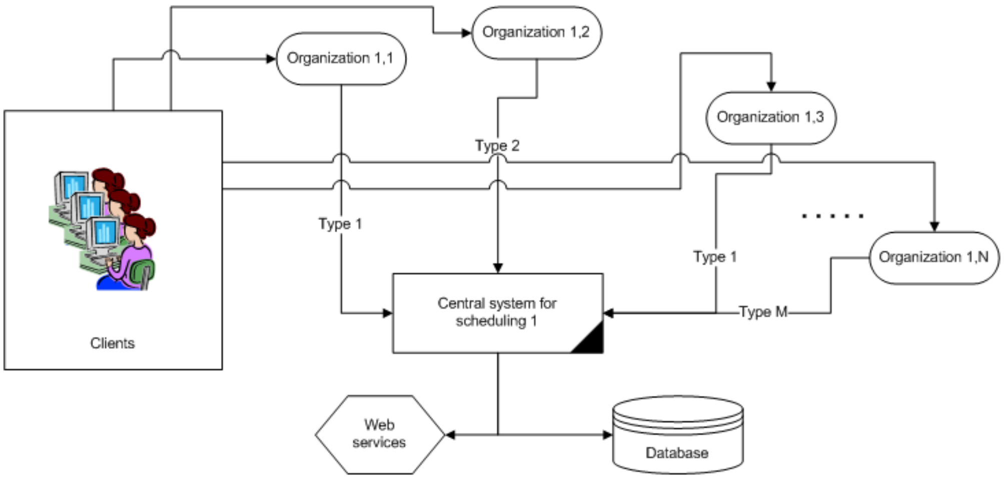

Figure 4. Architecture of a centralized scheduling system - traditional approach

In Figure 4 the structure of the aggregated scheduling system is presented when using the traditional approach without agents. We can identify several of the system's characteristics that are of interest to this study comparison:

1. If the user wants to make a comparison of the offers given by each organization for the available time slots, popularity, organization characteristics, etc. this task has to be performed manually by the user;
2. Inside the central system, the data search is key based. In case specific information is needed from the central system, there has to be a special purpose procedure that will make the extraction;
3. There is a high probability that data inconsistency will be present, especially in terminology;
4. The organization-to-organization communication would have to be implemented using specially devised web services, where the specified web services would have to be developed according to specific demands.

5.2. Multi-agent semantic aware approach

In order to provide seamless user friendly comparison of the different organization services (like searching for a free time slot across organizations, or checking the organization popularity) software agents can be used. In the following text we discuss the desing and analyse this type of solution. As mentioned previously, in order to provide communication between the agents there has to be a common language that the agents will understand. For these purposes an ontology has to be developed, and it can be used for searching time slots, making appointments, reviewing different statistics scores, etc. It is of extreme importance to reuse the standardized and already developed dictionaries and ontologies that offer a vast name space for all kind of descriptions, like for an example the FOAF (Friend of a friend) [12] RDF based scheme for describing personas and their socal connections, or the Dublin Core [13] that represents a dictionary for describing document properties.

In Figure 5 the agent based architecture approach for the same envisioned system is presented. The goal of the proposed multi-agent semantic aware system is to provide the user with aggregated information at one place. In order to provide its services, the system requires an agent for every organization, where the agent can mediate appointments and give information about the available time slots. There is an agent who will work for the client and according to the users desires it will ask for the requested time slot and give an offer to the user. If the user decides for one of the offered results the agent then needs to request an appointment schedule. The other agent who is engaged by the organization can live on the same or different platform. However, it needs to use the same ontology for content understanding and to comply with the FIPA specifications, so that it will be able to respond to the requests received from the client agent.

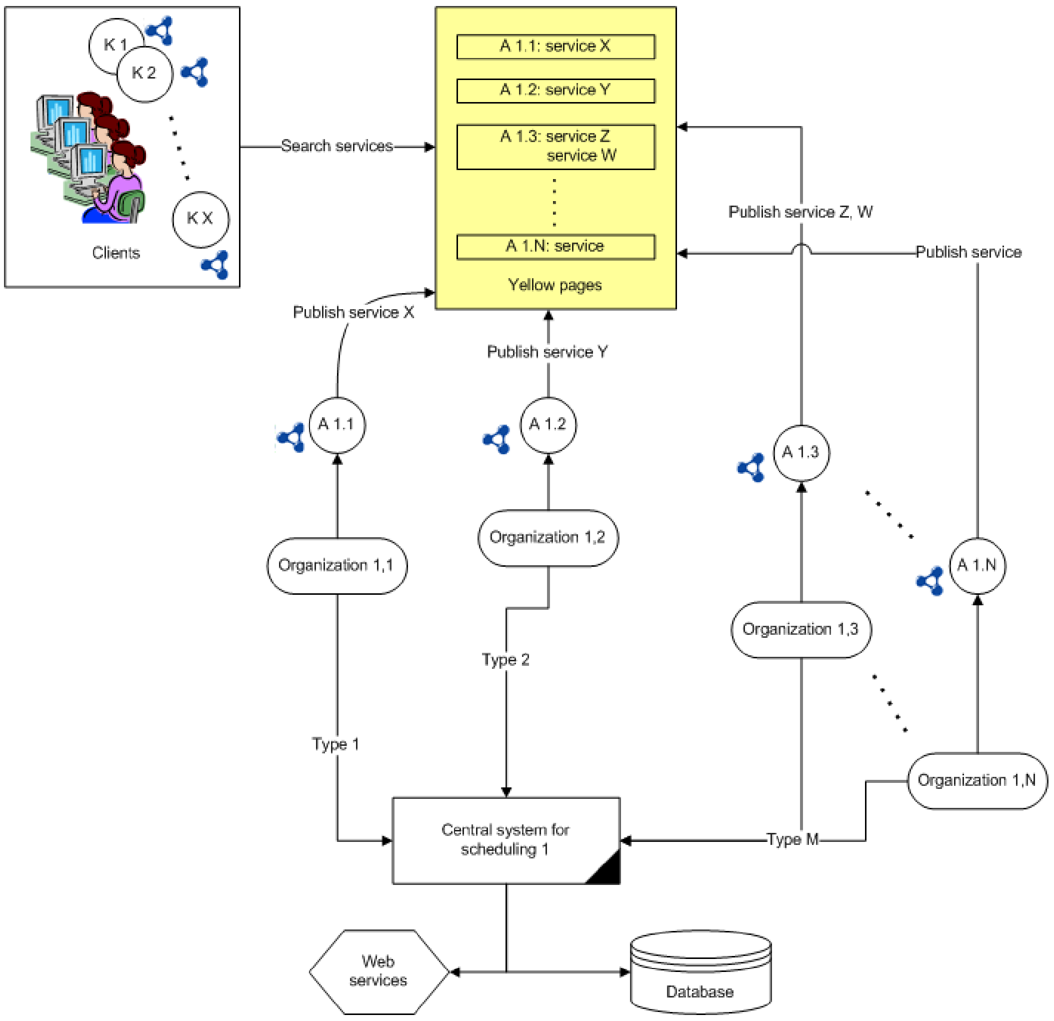

Figure 5. Scheduling system architecture using agents

In order for the user to be able to receive more valuable information (i.e. rating of the organizations, or similar data) an agent for running statistics is also introduced. In our case study, this agent takes into account the number of organized appointments made between different clients and organizations. The information that can be provided by this agent (like number of visits in organizations, last visits from given users (e.g. friends) etc.) can be used for possible advanced data filtering, i.e. give priority to businesses previously visited by the user. Because we are dealing with a multi-agent system, the time slot can be searched by a number of agents simultaneously.

All of these agents need to be registered in the directory of yellow pages with their corresponding service description. Using the information provided by the agents working in parallel, the user can go through the free time slots offered by more than one organization and pick the one that suits him best. The idea of the statistics agent is to improve the user experience and satisfaction and reduce the offer in quantity, while increasing the quality by checking ratings and taking into account comments from the other users, for example.

This type of architecture conformes to the FIPA specification discussed earlier in this paper. For the purposes of mutual agent understanding semantics is being used. This approach overcomes the downsides and limits of the seamless cross-organizational search for free time slots and similar characteristics. The given architecture can be additionally expanded with, for an example, a statistics agent that will be used for keeping summary statistics about the users satisfaction with the organization, comments, popularity, etc.

5.3. Multi-agent semantic aware application development

5.3.1. Requirements

The main goal is to develop a system for making appointments, which will make the appointment scheduling easier for the end-users, i.e. potential customers. In order to reach this goal the system need to provide the following:

1. User management – the user, which are intended to use the system, need to be able to make a simple login so that basic information can be used for the user in the system,
2. Maintaining agenda - each user has his personal agenda, which will reflect upon appointments made. Once a given user has successfully scheduled a meeting, all users must be informed and their agenda be updated accordingly.
3. Make appointment - the user can ask for an appointment for a specific date and time, and from the given results can chosse the free time slot and make the appointment.
4. Inquire for available time slots – a report for available dates, review of available dates at the most visited places and full report are example reports that can be returned as results to the user that wants to browse for an appointment.

5.3.2. Analysis and design

As a part of the higher-level steps while developing an application, like system specification or system goals identification, it’s necessary to use a comprehensive methodology that requires detailed information about the steps that are to be performed during development. There are many existing methodologies for designing software, wherein object-oriented analysis and design is quite extensively studied. This approach can be used in the design of an agent-oriented system, however it does not comply with the natural way of agents and the end results of the design are less likely to make efficient use of the agents.

As previously stated, one important aspect that agents have is that they are proactive, which means they keep to their agenda through a lifetime. This can be used as goal for the agents. The methodology that supports proactive agents needs explicit support for modeling purposes, and this is not generally part of the object-oriented methodologies. There are, however, several other methodologies that target the design of agent systems, such as: PASSI [14], MaSE [15], Tropos [16], Gaia [17], and Prometheus [18]. The number of agent methodologies is large, and this consistently confuses developers with the decision on which methodology to use. Care must be taken when choosing the appropriate tool, as only one methodology should be applied in order to design a real successful system. In this paper we decided to use the Prometheus Design Tool (PDT) [18] for the system design. This methodology consists of three stages: system specification (can be considered as pre-design), architecture design and detailed design, which provides the sufficient ground for our use case agent application described in this paper.

The system architecture mainly consists of three agent types:

1. Client agent - this agent should make requests for free time slots and present the obtained results to the client.
2. Organization agent – this agent’s service needs to be published in the yellow pages. The agent needs to be able to inform the client agent with the available time slots, and, if a client decided to make a schedule for a given time slot, the agent has to be able to make an appointment.
3. Statistics agent – this agent takes care for keeping the statistics information concerning the visits in different organizations by different users.

In Figure 6, the system agent architecture from global perspective is presented.

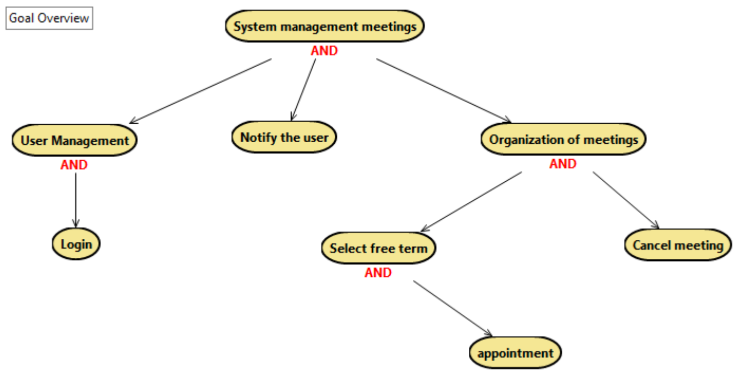

Figure 6. System goals for scheduling

In Figure 7, an overview of the agent system together with the exchange of messages between the agents is given.

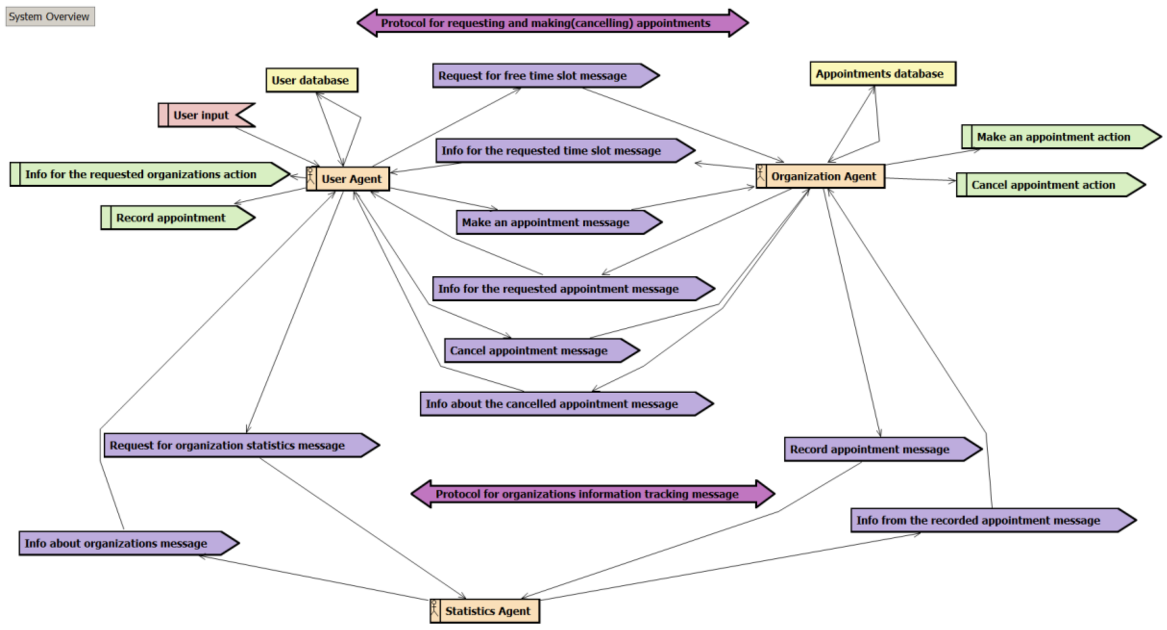
System Overview
Protocol for requesting and making(cancelling) appointments
User database
Request for free time slot message
Appointments database
User input
Make an appointment action
Info for the requested time slot message
Info for the requested organizations action
User Agent
Organization Agent
Cancel appointment action
Record appointment
Make an appointment message
Info for the requested appointment message
Cancel appointment message
Info about the cancelled appointment message
Request for organization statistics message
Record appointment message
Protocol for organizations information tracking message
Info from the recorded appointment message
Info about organizations message
Statistics Agent

Figure 7. Overview of the agent system and the exchange messages that provides inter-agent communication

5.3.3. Content language for the agent system

In order agents to be able to communicate, they have to use the same content language. Bean Generator [19], a plugin for Protégé, can be used for ontology Java classes creation. The generator is able to generate the classes needed to extend the simple abstract JADE ontology that contains the classes Concept, AgentAction which are the basic necessities for the classes generation. In Figure 8, the ontology for making the appointments, which extends the basic ontology for generating JADE classes, and the ontology for appointment statistics are given.

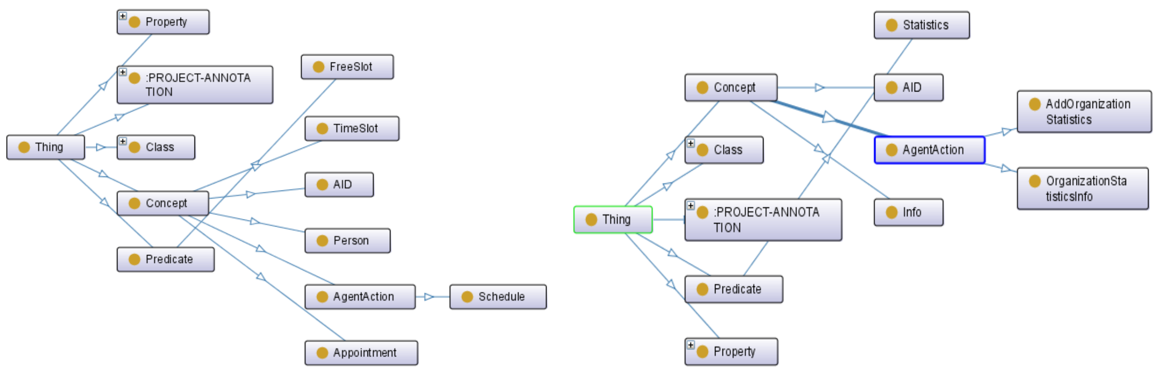

Figure 8. Ontology for appointment scheduling and statistics

5.3.4. Implementation of the user agent, agent of the organization and statistics agent

The user agent has four behaviours. The first behavior is the ability to request free time slots from the registered agents who are serving the selected service from the user. The second behavior is the request for info from the resulting organizations. The third behavior is scheduling a meeting and fourth behavior is the cancelling of already scheduled meeting. Classes for agent behavior are expanding the ContractNetInitiator class and AchiveREInitiator class.

The agent of the organization should be able to handle the demands of the other agents. On Figure 9 the sequence diagram for the inquiry of free time slots is presented.

5.4. Systems comparison and extension possibilities

The presented case study system can be extended by adding a price for each organization together with the possibility for negotiations, which will produce a lot more complex communication between the agents. This price negotiation scenario typically needs to use a lot more information that will allow the user to pick the best available offer. Some of this information can be processed automatically (including partial decision making) by agents, while for others a manual confirmation from the user or the organization may be required. If the negotiation agents have access to the, for an example, accounting system and the financial reports, they can reposnd much more efficiently.

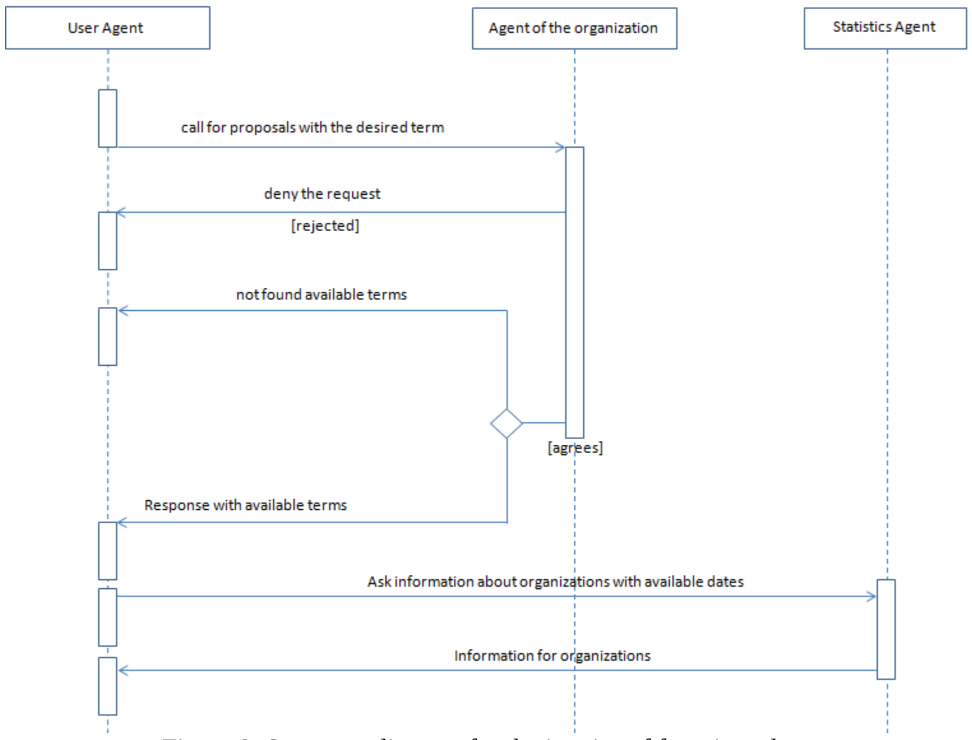
User Agent
Agent of the organization
Statistics Agent
call for proposals with the desired term
deny the request
[rejected]
not found available terms
[agrees]
Response with available terms
Ask information about organizations with available dates
Information for organizations

Figure 9. Sequence diagram for the inquiry of free time slots

Using the semantic aware multi-agent approach in system development, the system can very easily be extended or connected to another systems (for scheduling or other purposes) involving other organization. For this kind of an expanded integrated system to work, there is a necessity of one content language, which will be used by all of the involved agents. Thus, all of the agents need to comply with the FIPA specifications and the service description of the agents needs to be published in a directory, the so-called yellow pages discussed previously. This point only expands and intensifies the importance of using well defined standardizes ontologies when defining the agent communication since this will ensure the flexibility and future expandability of the system.

Another important remark is the usage of semantics for storing data based on RDF graphs. In the case when the central system traditional database is substituted with RDF graphs, a new powerfull characteristic is added to the system. In this way the system will be able to answer SPARQL questions, which would be a higher level search that offers a lot more possibilities compared to the standard key based search. At the same time, this solution will also not allow any room for data terminology inconsistency since all of the data will have inherent meaning.

The traditional system-to-system communication can be achieved using specifically created web services that need strictly defined input and output parameters. If there is a standardized dictionary that is adapted to the name space area for which the system is built (in our case, scheduling appointments of various types), the problem of strictly defined web services can be overcome. In this case, if a need arises to extract data from the system, a question will be given and the systems will be able to understand it correctly and reply in an adequate standardized format. Agents can be placed in each system in order to make the system-to-sytem communication even more complex.

The main benefit from using multi-agent semantic aware system compared to the traditional approach is the reduced involvement of the user together with the increased palette of services available for the end-user. The agent will inquire for an available time slot in multiple organizations at once. This means that the user agent is the one that needs to adapt to the environment, while the organization agents are offering more simple services. On the other hand, the organization agents need to be able to access the organization internal database in order to make the availability check or reschedule.

The question that our example case would like to answer is why use a software agent. In the today's web, there is more initiative into describing web services with semantics. This would be the place where agents can offer something more for the semantic web community. Agents have a lot in common with the web services. The agent architectures offer yellow pages, where agents can publish their service and where other agents can search for the needed service. However, there are crucial differences:

1. The web service is aware only of itself, not of his users or clients. Agents are often aware and by learning and modeling they are aware of other agents and their services also, so that inter-agent interaction can be achieved. This is one of the main features, since the lack of awareness does not permit future higher quality of experience for repetitive users because of lack of behavior adapting of the web service.
2. Web services are not designed to use ontologies unlike the agents. They can only use ontology to describe the service.
3. Agents are inherently communicative, while web servers are passive until awaken. Agents can provide alarms and updates when new information becomes available.
4. The web service is no autonomous unlike the agent. Agent authonomy is usually social authonomy, wherein the agents are aware of other agents and communicate with them, but keep their independence under certain circumstances.
5. Agents are cooperative and by forming teams and coallitions high level services can be offered.

Web services are extremely flexible and their main adgantage is that the web service developer does not need to know who or what would use the offered service. They can be used for internal information system interconnection in one organization, or for connecting the systems of multiple virtual organizations. However, the way web services can connect different systems will be based on technologies, which are developed for multi-agent systems.

6. CONCLUSION

The Semantic Web has a major role in the inter-agent communication enabling the agents to understand the complex information that passes between them. It enables the developers to manipulate the information using java or other object types, without additional transformations of the arrived information. Using semantics, agents that live on different FIPA platforms, but use same ontologies, can communicate, unlike serialization which allows for only inside platform communication in a non-human format. Also, using the XML technologies, developers can define structures that can be shared by the agents in a multi-agent system. In the end, what the semantics can offer is enabling the agent to perform much more complex tasks.

Multi-agent systems can become the fundamental blocks of software systems, even in the cases when software systems have no need of agent behavior. For a given conventional system constructed with agents as a module based application, the following characteristics will be present: (1) The more agent-based modules are active, better representation of real things. (2) Modules can keep their promises. (3) Negotiations between the modules, over social networks, collaboration and opinion can be exchanged in order to obtain the results. (4) Modules can be aggregated into a part of a more complex software system.

Meanwhile, by using semantic agents to create software agents we have the following benefits: (1) Agents allow dynamic composition when the component of the system may be unknown by the time of execution. (2) By using agents abstract interactions are allowed, which means that by the time of their execution they can be unknown. (3) Since the agents can be added and removed in the system, the system software can be changed during its lifetime cycle. (4) Agent can represent many views and they can use many different procedures for decision making, so in a way they can produce much more robust software.

From this experiment, it’s never easy to create robust software. If the development process was approached with the paradigm of creating semantic agents, the developers would need to take into consideration the following remarks: (1) Algorithms in form of agents are easy for reuse, since agents are easy to be added, especially to an existing system, and the agents have the ability to interact with many other agents. (2) In order to make a full usage of the agents, organization specifications need to be developed. (3) In order to establish a full understanding between the agents they should be able to understand each other, as they should be able to detect and fix inconsistency without a leader. (4) When agents use non-renewable resources there could be a raise of issues whether there is limited processing cycles, limited memory, power, network bandwidth, since they would use it *N* times faster.

ACKNOWLEDGEMENTS

This work has been partially supported by the EUROWEB project (http://www.mrtc.mdh.se/euroweb) funded by the Erasmus Mundus Action II programme of the European Commission.

REFERENCES

[1] Shoham, Y.; , “An overview of agent-oriented programming”, *Software agents*, pp. 271-290, MIT Press Cambridge, MA, USA ©1997

[2] Tweedale, J. W.; Jain, L.C.; , “Agent Oriented Programming”, *Embedded Automation in Human-Agent Environment Adaptation, Learning, and Optimization*, Springer, Vol. 10, pp. 105-124, 2012

[3] Daconta, M.C.; Obrst, L.J.; Smith, K.T.; , *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, Wiley & Sons, 2003

[4] Shadbolt, N. R.; Berners-Lee, T.; , “The Semantic Web Revisited”, *Intelligent Systems, IEEE*, Vol. 21, No. 3, pp. 96-101, 2006

[5] Bordini, R.H.; *Multi-Agent Programming: Languages, Tools and Applications*, Springer, 2009

[6] History of FIPA. [Online] Available at: http://www.fipa.org/subgroups/ROFS-SG-docs/History-of-FIPA.htm

[7] FIPA Agent Mangement. [Online] Available at: http://www.fipa.org/specs/fipa00023/SC00023K.html

[8] Antoniou, G.; van Harmelen, F.; *A Semantic Web Primer*, 2nd Ed., 2008

[9] Hendler, J.; , “Agents and the Semantic Web”, *Intelligent Systems, IEEE*, Vol. 16, No. 2, pp. 30-37, April 2011.

[10] García-Sáncheza, F.; Valencia-Garcíaa, R.; Martínez-Béjarb, R., Fernández-Breisa, J.T.; , “An ontology, intelligent agent-based framework for the provision of semantic web services”, *Expert Systems with Applications* Vol. 36, No. 2, Part 2, pp. 3167–3187, March 2009

[11] Caire, G.; Cabanilas, D.; “Jade Tutorial Application – Defined Content Languages and Ontologies”. [Online] Available at: http://jade.tilab.com/doc/tutorials/CLOntoSupport.

[12] FOAF – Project for creating a Web of machine readable pages, describing people. http://semanticweb.org/wiki/FOAF

[13] Dublin Core Metadata Initiative, http://dublincore.org/

[14] Burrafato, P., Cossentino, M.; , “Designing a multi-agent solution for a bookstore with the PASSI methodology”, *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*, Toronto, 2002

[15] DeLoach, S.A.; , “Analysis and design using MaSE and agentTool”, *Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001)*, 2001

[16] Bresciani, P.; Perini, A.; Giorgini, P.; Giunchiglia, F.; Mylopoulos, J.; “Tropos: An agentoriented software development methodology”, *Autonomous Agents and Multi Agent Systems* 8(3) pp. 203–236, May 2004

[17] Zambonelli, F.; Jennings, N.; Wooldridge, M.; Developing multiagent systems: the gaia methodology. ACM Transactions on Software Engineering and Methodology 12(3) (2003)

[18] Padgham, L.; Thangarajah, J.; Winikoff, M.; “Tool support for agent development using the prometheus methodology”, Cai, K.Y., Ohnishi, A., Lau, M., eds.: *Proceedings of the Fifth International Conference on Quality Software (QSIC 2005)*. Workshop on Integration of Software Engineering and Agent Technology (ISEAT). pp. 383–388, Sep 2005

[19] Protégé, http://protege.stanford.edu/

[20] Tariq Mehmood, M. Shahid Farid; “A Multi-Agent Architecture for Task Scheduling In University Environment”, IAES International Journal of Artificial Intelligence (IJ-AI); Vol. 1, No. 4, Dec 2012, pp. 193~200; ISSN: 2252-8938

[21] Guohai Zhang, Yusheng Li; “Multiple Disciplines Product Design Knowledge Representation Strategy Based on Ontology and Semantic Network”, TELKOMNIKA, Vol. 11, No. 10, October 2013, pp. 6074 ~ 6079; e-ISSN: 2087-278X

BIBLIOGRAPHY OF AUTHORS

Trajche Manev was born on 8th of March 1985. He received the B.S degree in Electrical Engineering and Information Technologies from the Ss. Cyril and Methodius University in Skopje, Macedonia 2009. M.S degree in Computer Science and Engineering from the Ss. Cyril and Methodius University in Skopje, Macedonia 2013. He currently is dealing with professional software development in the IT company 6PM.

Sonja Filiposka is born in Skopje in 1980. She received her B.Sc. degree in Electrical Engineering in 2003, M.Sc. in Electrical Engineering in 2006 and PhD in Technical Sciences in 2009 at the Ss. Cyril and Methodius University in Skopje, Macedonia. She works as an Assistant Professor at the Faculty of Computer Science and Engineering. She is an author of over 60 scientific papers published in international conference proceedings and journals and is an active participant of a number of national and international research projects. Currently she is on her PostDoc stay at the University of the Balearic Islands with the group of Computer Architecture, Performances and Communications.