Business Intelligence Seeker - User Agent

Project Acronym	UNDERSTANDER
Document-Id	D.3
File name	
Version	Final document
Date	start: 01 January 2014 end: 28 February 2014
Author(s)	Violeta Damjanovic (SRFG)
QA Process	Verteiler: Prüfung durch: Genehmigung durch:

Table of Contents

1. Introduction

The following table summarizes the main goals, content, methods and milestones related to Work Package 3 (WP3) of *UNDERSTANDER*.

Goals	The main goal of this report is to develop the BI agent that can be "primed" with the BI scripts, gets a seed set of URIs, then starts aggregating BI knowledge by traversing the WWW from the seed URIs. Alternatively, we will develop a variant that can be seeded via the search result from one of the search engines.
Description of the Content	● Software of the web crawling ● BI seeking agent
Method	● Semi-formal specification and subsequent programming in JADEX and Prolog via RESTful services
Milestones	● User agent with domain-specific priming scripts for BI ● Prototype search service following the Russell & Norvig's agent methodology, with the agent's evaluation function specialized through the formalized and customized BI scripts ● Open source software + specification + paper

1.1. Motivation

User agents are characterized by holding different views of the world, which requires a common ontology to be used to allow for their interoperability and cooperation, without affecting their autonomy. In other words, user agents need to communicate and share an agreed terminology describing a certain domain of discourse. This terminology is considered to be the common domain ontology. However, as the authors in (Malucelli, 2006) noticed: "*even with a common domain ontology, people may use different terms to represent the same item, or choose a more general, or detailed representation.*" Although some authors believe this can be overcome through ontology integration, ontology alignment, or by merging two ontologies (c.f. (Pinto et al., 1999)), recent discussion on the role of ontologies versus Linked Data brings new perspective for knowledge share (Nikolov & Motta, 2010) (Ding, Finin, & McGuinness, 2010).

At the same time, the growth of new distributed computing standards became a critical driver for the development of next generation systems. In that context, the role of complex agent systems is expected to bridge the gap between research and applications more than even before, by incorporating non-trivial agent-based simulations tools into everyday systems engineering.

1.2. Scope

User agents in UNDERSTANDER are developed in JADE multi-agent platform. This report discusses the relation between the agents and their knowledge base (previously presented in D.4 "Business Intelligence Knowledge base"), the agent communication protocol and specific behaviour implementing their functionality.

1.3. Structure of the Document

Section 2 discusses the state of the art technologies in several areas related to Web agents, such as Multi-Agent Systems (MASs), ontologies used by MASs, interaction and simulation based on MASs. Section 3 presents agent's communication protocols in UNDERSTANDER, as well as their specific behaviour, which determines the way of interaction between the Client and the Server agent. Section 4 concludes the document.

2. Related Work

This chapter presents the state of the art in several areas related to Web agents, such as Multi-Agent Systems (MASs), ontologies used by MASs, interaction and simulation based on MASs.

2.1 Agents and Multi-Agent Systems

The authors in (Wooldridge & Jennings, 1995) present one of early discussions on Web agent technologies. (Nwana, 1996), another early reference paper on agent technologies, gives an overview of the different agent types such as: collaborative agents, interface agents, mobile, information, reactive, hybrid, heterogeneous, smart agents. (Wooldridge, 2002) defines an agent as "*a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives*". It is a computer program with a relatively complete functionality, which cooperates with others to meet its designed objectives (Qingning et al., 2003). An agent can act in a flexible and autonomous way, within the environment where it is situated (Jennings, 2000). It is task-oriented and capable of decision-making (Marivate et al., 2008).

Although agents can act separately to solve a particular problem, it is frequent for systems to be composed of several agents developed to cooperate in a complex problem, involving data, knowledge or distributed control (Oliveira et al., 1999). A composition of several agents with the capability of mutual interaction and communication in order to achieve goals, is known as **Multi-Agent Systems** (MAS). It is used to solve complex problem that cannot be done by individual agent; for example, complex problems such as distributed domains (e.g. global manufacturing supply chain network (Jiao et al., 2006) (Gog & Gan, 2005)), distributed computing (Zhong et al., 2004) (Chira, 2007), software collaborative development environment (Ahamo & Aljawaherry, 2012) (Chuan, 2011), etc.

A MAS is usually designed and developed in a modular fashion to cover various points of view (expert's knowledge), cooperate (interact) through a set of actions, and be reusable. Interaction between agents is expected to be reactive, and influence the current status and the future behaviour of the agents. "*The agents interact through a series of events, during which they are in contact with each other in some way, whether this contact is direct or takes place through another agent or through the environment*" (Ferber, 1999).

A MAS has ability to increase the efficiency and effectiveness of working groups in distributed environments. For example, the authors in (Romero et al., 2008) introduced a MAS-based simulation tool to support training in global **requirement elicitation process**. They used agent technology to simulate various stakeholders in order to enable requirement engineers to understand and gain experience in acquiring requirement elicitation. Another example is Col_Req, the multi-agent based collaborative requirements tool that supports **requirement engineers for real-time systems** during the requirement engineering phase (Giri, 2011). The authors in (Pakdeetrakulwong & Wongthongtham, 2013) proposed a conceptual framework of **a MAS-based recommender system** to provide active support to access and utilize knowledge and project information (based on the software engineering ontology) for long-distance collaborative work (e.g., distributive (multi-site) development of software systems).

Nowadays, Agent-based Directed Simulation (ADS) is an area exploring agent models for development of the domain-specific simulations, simulation techniques and toolkits. ADS can be classified into two categories: (i) agent-supported simulation (the use of agents as a support facility to enable computer assistance in problem solving), and (ii) agent-based simulation (the use of agents for the generation of model behaviour in a simulation). Section 2.4 discusses MAS-based simulation in more details.

2.2 Ontologies in Multi-Agent Systems

Knowledge sharing and exchange is one of the key factors in the development of MAS (Iordan et al., 2008). Each agent need to collaborate with other agents, which implies their ability to communicate and understand messages from one another. Agent communication requires the use of communication protocols, communication languages and ontologies. Ontologies play an important role as they can support the integration of heterogeneous and distributed information sources.

Lack of standardization, which hampers communication and collaboration between agents, is known as the interoperability problem (Willmott, 2001). In that context, ontologies can be used to facilitate the semantic interoperability, while Agent Communication Language (ACL) defined by FIPA can be used as the language of communication between agents.

The authors in (Pakdeetrakulwong & Wongthongtham, 2013) identified several research direction that integrate the use of ontologies and MAS. Similarly, the authors in (Paydar & Kahani, 2011) introduced a multi-agent framework for automated **testing of web-based applications**. (Lee & Wang, 2009) proposed an ontology-based computational intelligent MAS for **Capability Maturity Model Integration (CMMI) assessment**. The authors developed the CMMI ontology to represent the CMMI domain knowledge. The research presented in (Nunes et al., 2011) addresses the integration of MASs and **Software Product Lines (SPLs)**. The authors created an ontology for modeling the Multi-Agent System Product Lines (MAS-PLs). MADIS (Chira, 2007) is a multi-agent design information system developed to support the **distributed design process**. The MADIS ontology was developed to formally conceptualize the engineering design domain and enable knowledge sharing, reuse and integration in a distributed design environment. The authors in (Monte-Alto et al., 2012) proposed a multi-agent **context processing mechanism** called ContextP-GSD (Context Processing on Global Software Development) that utilizes contextual information to assist user's task during the software development project. OntoDiSen is an application ontology exploited in the ContextP-GSD system, representing GSD contextual information.

In his PhD thesis, Malucelli (Malucelli, 2006) investigates ontology-based **services for agents**

interoperability. In that context, Malucelli compared various approaches to support communication among agents using different ontologies. Here, we only briefly summarize the state-of-the-art subsection presented in (Malucelli, 2006): "*In the case of MAS, FIPA has identified and analysed different types of interoperability problems that arise and has, consequently, proposed the creation of an* ***Ontology Agent (OA)*** *to assist the community of agents in the alignment of ontologies. However, the implementation of such service is left to system developers. Furthermore, the FIPA Ontology Service Specification classifies this domain-dependent task as very complex and possibly not always achievable. An implementation of the OA is presented in (Suguri et al., 2001), which is a sample application of an ontology shopping service that integrates multiple database schemata to verify and demonstrate the specification. However, no mechanism is provided to match terms between different ontologies.*"

Only in recent years, the problem of handling different ontologies in MAS has been addressed again. (Malucelli, 2006) gives a summary of the main contributions in this domain, as follows:

- (Steels, 1998) proposes a complex adaptive system approach based on an ontology and a shared lexicon in a group of distributed agents, which are characterized by local interactions, without central controlling agency. An agent can associate a single word with several meanings and a given meaning with several words. The words are matched using distance measurements.
- (Bailin & Truszkowski, 2002) describes an approach to ontology negotiation that allows Web-based information agents to resolve mismatches in real-time, without human intervention. The system employs the WordNet database as a data source to extend each ontology's concept repertoire.
- (van Eijk et al., 2001) develops a communication mechanism in which translators between the vocabularies of agents are generated. These translators are dynamically constructed during the execution of the system and are based on the information the agents exchange and on their underpinning ontologies. In this approach, there is no global shared ontology and each agent has its own private ontology.
- (Tzitzikas & Meghini, 2003) considers peer-to-peer systems in which peers employ taxonomies for describing the content of their objects and formulating semantic-based queries to the other peers of the system. Each peer uses its own taxonomy and is equipped with inter-taxonomy mappings to carry out the required translation tasks. This methodology does not make any assumptions on how the involved taxonomies are constructed or how they are used, but it requires the presence of two databases that contain several common objects.
- (Burnstein et al., 2003) has sketched an approach to automatic derivation of programs for translating the output of a source agent to the input representation of a target agent, based on lambda-calculus. However, the approach does not provide any method for establishing a mapping between heterogeneous ontologies.
- (Doherty et al., 2005) combine logic-based techniques with approximate reasoning. It provides software or robotic agents with the ability to ask each other approximate questions concerning unclear or unknown terms and actions. Each agent can communicate in the language of the other agents and has a mediation function.

- (Williams et al., 2003) proposes a methodology for agents to develop local consensual ontologies as a means supporting the communication within a multi-agent system of B2B agents. The agents need to find related services (ontology concepts) between inter-organisation ontologies and intra-organisation ontologies.
- (Wiesman and Roos, 2004) proposes a domain-independent methodology for handling interoperability problems by learning mappings between ontologies. The learning method is based on exchanging instances of concepts defined in the ontologies. They focus on establishing a mapping between two concepts, one from each ontology.
- (van Diggelen et al., 2005) addresses the problem of establishing a suitable communication vocabulary in a formal and abstract way. Each agent has a private ontology which is incomprehensible to the other agents. The set of shared concepts is represented in the communication vocabulary. To preserve soundness, the agents translate (adopting a distribution function) private concepts into equivalent or more general shared concepts.

Finally, approach proposed by (Malucelli and Oliveira, 2006) is focused on the **resolution of negotiation conflicts in a B2B domain**. For that purpose, authors defined a set of services addressing the interoperability problems during inter-agent communication. They use a mediator agent called OSAg, which is responsible for the resolution of all negotiation conflicts that occur within the MAS. All the matched concepts are memorised by the OSAg and kept for the future negotiation rounds. The mapping between ontologies is established by comparing, for each pair of concepts, the attributes (grouped by data type), the relation *has-part* and the descriptions of the concepts.

2.3 Strategic Interaction in Multi-Agent Systems

(Panait & Luke, 2005) analyses interaction in MASs in terms of game-theory; for example, agent interaction as a form of strategy games consisting of matrices of payoffs for each agent, based on their joint actions. For example, evolutionary game theory was successfully used to study the properties of cooperative coevolution (Ficici & Pollack, 2000) (Wiegand, 2003), to visualize basins of attraction to Nash equilibria for cooperative coevolution (Panait et al., 2004), etc. In the area of coordination games, various repeated games are introduced in the literature to highlight specific issues associated with MAS. Claus and Boutilier (Claus & Boutilier, 1998) introduce two simple 3x3 matrix games:

- a coordination game with two optima and high penalties for mis-coordination; and
- a coordination game with two Nash-equilibrium points, one of them corresponding to a suboptimal collaboration.

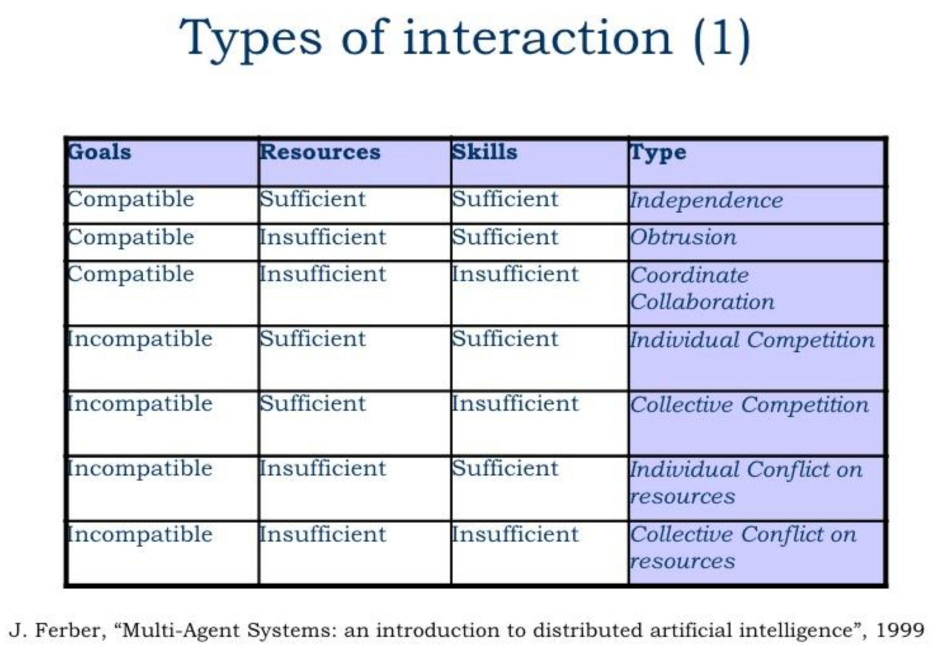
These games are later used in (Kapetanakis & Kudenko, 2002) (Lauer & Riedmiller, 2000) (Panait & Wiegand, 2003) to investigate multi-agent reinforcement learning and evolutionary computation approaches.

Social dilemmas in game-theory concern the individual decisions of several agents, all of which receive a joint reward (Glance & Huberman, 1994). The *Iterated Prisoners' Dilemma*, *Tragedy of* *the Commons*, *Braess Paradox* and *Santa Fe Bar* are some examples of social dilemma games (Panait & Luke, 2005):

- The *Iterated Prisoner's Dilemma* involves two or more agents supposedly accused of a robbery; the agents have to choose between two actions: confess the crime or deny participation in it. The settings are such that it is rational for individual agents to deny, but it is in their collective interest for all to confess.
- In the *Tragedy of the Commons*, a number of agents share a resource of limited capacity. When the joint usage of the resource exceeds the capacity, the service deteriorates, and so do the rewards received by the agents.
- In the *Braess Paradox problem*, agents share two resources. The dilemma arises when agents need to decide to start accessing the less utilized resource: if all agents decide to do so, it will become overwhelmed and rewards will drop. Further details on these problems, accompanied by a co-evolutionary approach to learning solutions to them, can be found in (Mundhe & Sen, 2000).
- In the *Santa Fe Bar problem*, a large number of agents individually must decide whether to go to a bar in Santa Fe. If too many or too few agents opt to go, their satisfaction is lower than when a reasonable number of them decide to go (Arthur, 1994) (Greenwald et al., 2002) (Wolpert et al., 1999).

Social dilemma problems have been used to model practical issues in real multi-agent problems, such as e.g., network routing. During interaction, agents are in contact with each other either directly, through another agent, or through the environment. (Ferber, 1999) classifies the following types of interaction (see Table x) (Table below found in (Manzoni, 2009)):

- **Independance**: a simple juxtaposition of actions carried out by agent independently without effective collaboration;
- **Obstruction**: agents get in touch in accomplishing their tasks, but they do not need one another;
- **Coordinate collaboration**: agents have to coordinate their actions to have synergistic advantages of pooled skiles (e.g. industrial activities);
- **Individual competition**: resources are not limited and the competition is not related to them;
- **Collective competition**: agents have to group into coalitions or associations to be able to achieve their goals;
- **Individual conflict on resources**: the object of conflict is insufficient resource;
- **Collective conflict on resources**: all forms of collective conflicts in which the objective is to obtain possession of territory or a resource.



Types of interaction (1)

Goals	**Resources**	**Skills**	**Type**
Compatible	Sufficient	Sufficient	*Independence*
Compatible	Insufficient	Sufficient	*Obtrusion*
Compatible	Insufficient	Insufficient	*Coordinate Collaboration*
Incompatible	Sufficient	Sufficient	*Individual Competition*
Incompatible	Sufficient	Insufficient	*Collective Competition*
Incompatible	Insufficient	Sufficient	*Individual Conflict on resources*
Incompatible	Insufficient	Insufficient	*Collective Conflict on resources*

J. Ferber, "Multi-Agent Systems: an introduction to distributed artificial intelligence", 1999

Interaction models in MASs are often inspired by other disciplines, e.g. social science, biology, etc. These models can support either direct, or indirect communication. For example, KQML (Knowledge Query and Manipulation Language) and KIF (Knowledge Interchange Format) are examples of direct communication languages. KQML defines performatives to support conversation among agents; KIF allows to represent knowledge and information about agents, beliefs, desires, intentions, perception plans. In addition, agents must share an ontology to describe a domain. In indirect agent communication, agents interact through an intermediate entity with the access rules and interaction mechanisms.

2.4 Multi-Agent System Based Simulation

Agent-oriented approaches have also had an effect on simulation methods (Yilmaz & Ören, 2009a). (Uhrmacher, 2002) shows several new challenges in the context of evaluating software agents by simulation-based approaches. A prominent example is documented in (Himmelspach et al., 2003), that is the **synchronization problem of simulation** software implementing the testbed and the software agents under test. **Asynchronous interaction** provides a loose coupling between simulation and agents, as shown in RoboCup and RoboCup Rescue scenarios (Takahashi, 2008).

Another effect that agent research has had on modeling and simulation methods has been in the context of **model composition and simulation interoperation** (Tolk, 2006) (Yilmaz and Tolk, 2006). Current simulation protocols are focused on the definition of standardized information exchange, such as Protocol Data Units in IEEE1278 (of Electrical and Engineers) or Federation Object Model in IEEE1516 (of Electrical and Engineers). The matching of simulation internal data to these information exchange elements is typically hard-coded. By supporting reuse at the modeling level, other agents model have become of interest, such as meta-description and ontologies for selecting suitable models and for relating different modeling formalisms (Tolk et al., 2007).

The synergy of simulation and software agents is the essence of a novel research area, called **Agent-Directed Simulation (ADS)**. ADS opens new vistas, and has important practical implications (Yilmaz & Ören, 2009a) (Yilmaz & Ören, 2009b) (Ören & Yilmaz, 2012). The emergent need to model complex situations whose overall structures emerge from interactions between individual entities and cause structures on the macro level to emerge from the models at the micro-level, is making agent paradigm a critical enabler in modeling and simulation of complex systems.

Two overall aspects of simulation are (i) experiments and (ii) experience (Ören & Yilmaz, 2012). From the perspective of experiments, simulation is about performing goal-directed experiments using models of dynamic systems. Experiments are performed for decision support, understanding, and education. From the point of view of experience, simulation is about gaining experience by the use of a representation (or a model) of a system. Gaining experience through simulation could be done for two categories of activities, e.g. for training and for entertainment (simulation games).

ADS includes contributions of **simulation to agents** (i.e., agent simulation) and contributions of **agents to simulation** (i.e., agent-supported simulation and agent-based simulation).

- **Agent-supported simulation** is the use of agents as a support facility to (1) enable computer assistance in problem solving, or (2) enhance cognitive capabilities of simulation systems. As support facility, agents can support front-end user system interface functions, such as problem specification, or back-end user-system interface functions, such as data compression, explanation, problem and/or solution documentation, and solution selection. Agents can enhance cognitive capabilities of modeling and simulation systems by providing understanding and multi-understanding abilities. About 60 types of machine understanding are explained in (Ören, 2000) and (Ören et al., 2007). Multi-understanding and switchable understanding are explained in (Ören et al., 2009).
- **Agent-based simulation** focuses on the use of agents for the generation of model behavior in a simulation study, such as dynamic model composition while simulation is running. In (Ören & Yilmaz, 2009a), authors elaborate on agent-directed simulation, as well as three annual events and associated publications on ADS.

3. User Agents in UNDERSTANDER

UNDERSTANDER's user agents are based on Russell and Norvig's model of a **Utility-based Agent** (see Figure 1 below), which tries to maximize its own "happiness". The Utility-based Agent differs from Goal-based Agent by adding a utility measure, that is a function specifically applied to the different possible actions that can be performed in the environment. The Utility-based Agent rates each scenario to see how well it achieves certain criteria with regard to the production of a good outcome (Mills & Stufflebeam, 2005). Things like **the probability of success, the resources needed to execute the scenario, the importance of the goal to be achieved, the time it will take**, might all be factored into the utility function calculations.

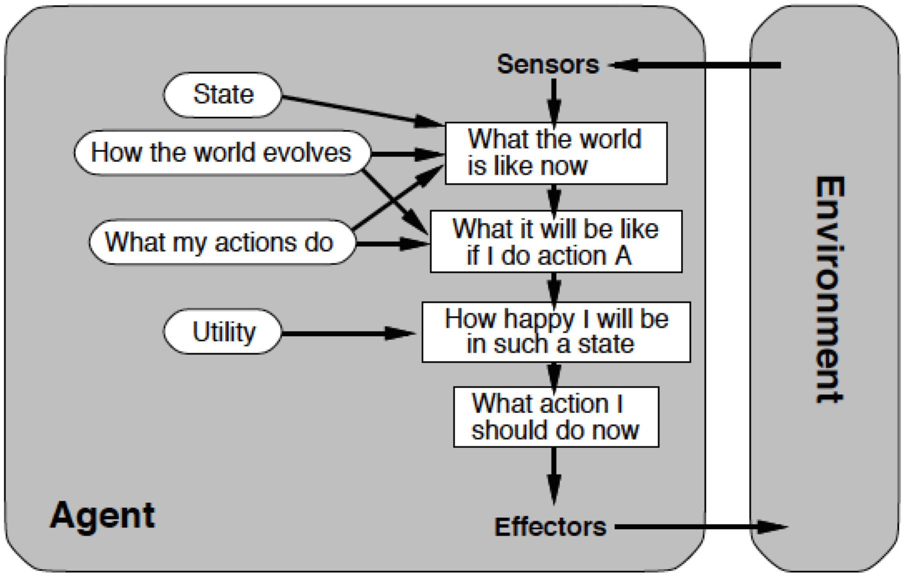

Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Effectors
Agent
Environment

Figure 1. A Complete Utility-based Agent (Source: http://www.cs.berkeley.edu/~russell/aima1e/chapter02.pdf)

Intelligent agents maximize their utility functions that proactively pursue their goals. Apart knowledge about the world, which makes the agent autonomous, they also need some knowledge on their *percept sequence* (Mills & Stufflebeam, 2005). It is not always predictable, and depends on the constantly changing world (environment), which further influence mapping of decision procedure to a plan of action in pursuit of its goals. Since the programmer cannot generally predict every state of the world that will be confronted by the agent, by giving the agent some goals, the ability to constantly reassess its situation, the ability to learn through trial and error, and in addition giving it a number of plans and ways of evaluating those plans as they become possible paths to the goal, the agent gets an enormous amount of flexibility and adaptability. To achieve such a functionality, M. Wooldridge proposes the following basic control loop of an autonomous agent (Wooldridge, 2002):

```
while true
       observe the world;
       update internal world model;
       deliberate about what intention to achieve;
       use means/ends reasoning to get a plan for the intention
       execute the plan
end while
```

In other words, the agent observes the world and collects percepts. The agent updates its internal world model by adding the new percept to its percept sequence and pre-programmed information about the world. Deliberation about what intention to achieve, given the updated world model, is based on the overall goals of the agent. Once a decision is made about what intention to achieve, the agent consults its plan library and/or its decision procedures (e.g., means/ends reasoning) for determining what means to use to reach its end. Finally, the agent executes the plan, provided no new percept calls for an altering of its current intention. By adding the agent's ability to learn from interacting with other agents, human and computers, to the above model, the flexibility and adaptability of the agent will only improve.

3.1 A Communication Protocol in UNDERSTANDER

The agent communication in *UNDERSTANDER* follows a simple protocol, as described and illustrated below, in Figures 2-3:

- To make an operation, the *ClientAgent* sends a **REQUEST** message to the *ServerAgent*. The *ServerAgent* responds with an **INFORM** after processing the request, or with an **NOT_UNDERSTOOD** message if it cannot decode the content of the message.

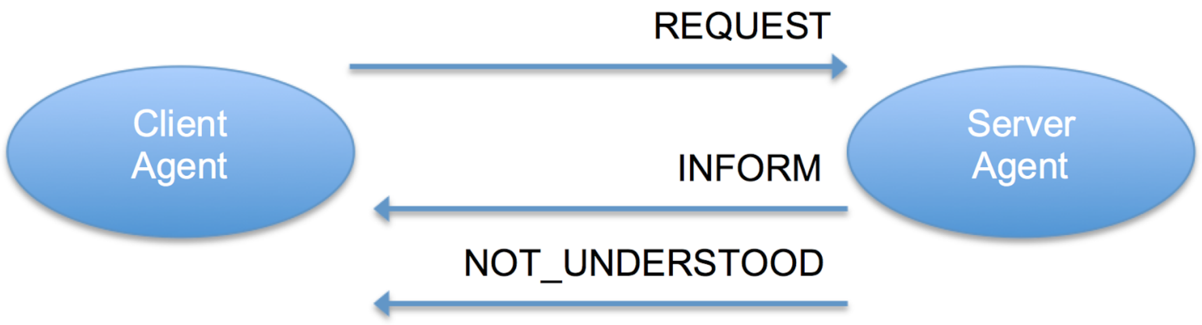


Figure 2. The agent communication protocols in *UNDERSTANDER*: REQUEST

- To query specific information, the *ClientAgent* sends a **QUERY_REF** to the *ServerAgent*. The *ServerAgent* responds with an **INFORM** message after processing the query, or with a **NOT_UNDERSTOOD,** if it cannot decode the content of the message.

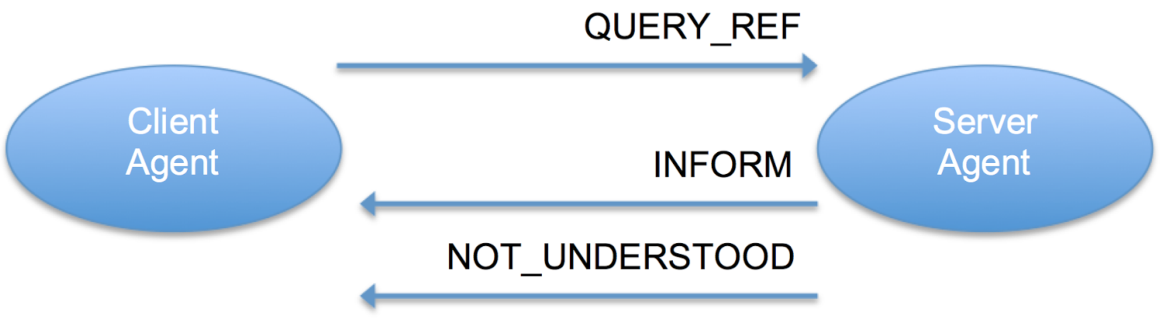


Figure 3. The agent communication protocols in *UNDERSTANDER*: QUERY_REF

3.2 Agent Behaviour

The agents in MAS operate independently and in parallel with others agents. Agent's parallelisms could be implemented by assigning a Java Thread to each agent, which is rather slow and not very efficient in case of large-scale parallelism. Therefore, to support efficiently parallel activities within an agent, JADE (c.f. http://jade.tilab.com/) has introduced a concept called **Behaviour**.

Behaviour is an *Event Handler*, a method which describes how an agent reacts to an *event* (JADE, 2004). In JADE, behaviours are defined as classes and the *Event Handler* code is placed in a method called ***action***. For example, coding a negotiation process includes the following steps: (i) sending offer, (ii) waiting for counter-offers, and (iii) reaching an agreement. This activity consists of an alternation of *active phases* (when the agent decides what to do and sends messages), and *passive phases* (when the agent waits for an answer). Each behaviour execution corresponds to one single instantaneous active phase. To implement long-term activities like a negotiation, we have to provide as many different **Behaviours** as there are *active phases* in the activity (one for every active phase). We must also arrange for them to be created and triggered in the right sequence; for example, by specifying behaviour scheduling introducing time parameters (defined in milliseconds).

JADE provides various **Behaviours** which can be extended to model the complex activity of real agents. In general, there exist two kinds of behaviour classes:

- **Primitive Behaviours**, such as the *Simple* or *Cyclic* Behaviours, and
- **Composite Behaviours,** which can combine both simple and composite behaviours to be executed either in sequence or in parallel.

Figure 4 shows an annotated UML class diagram for JADE behaviour (JADE Guide, 2010).

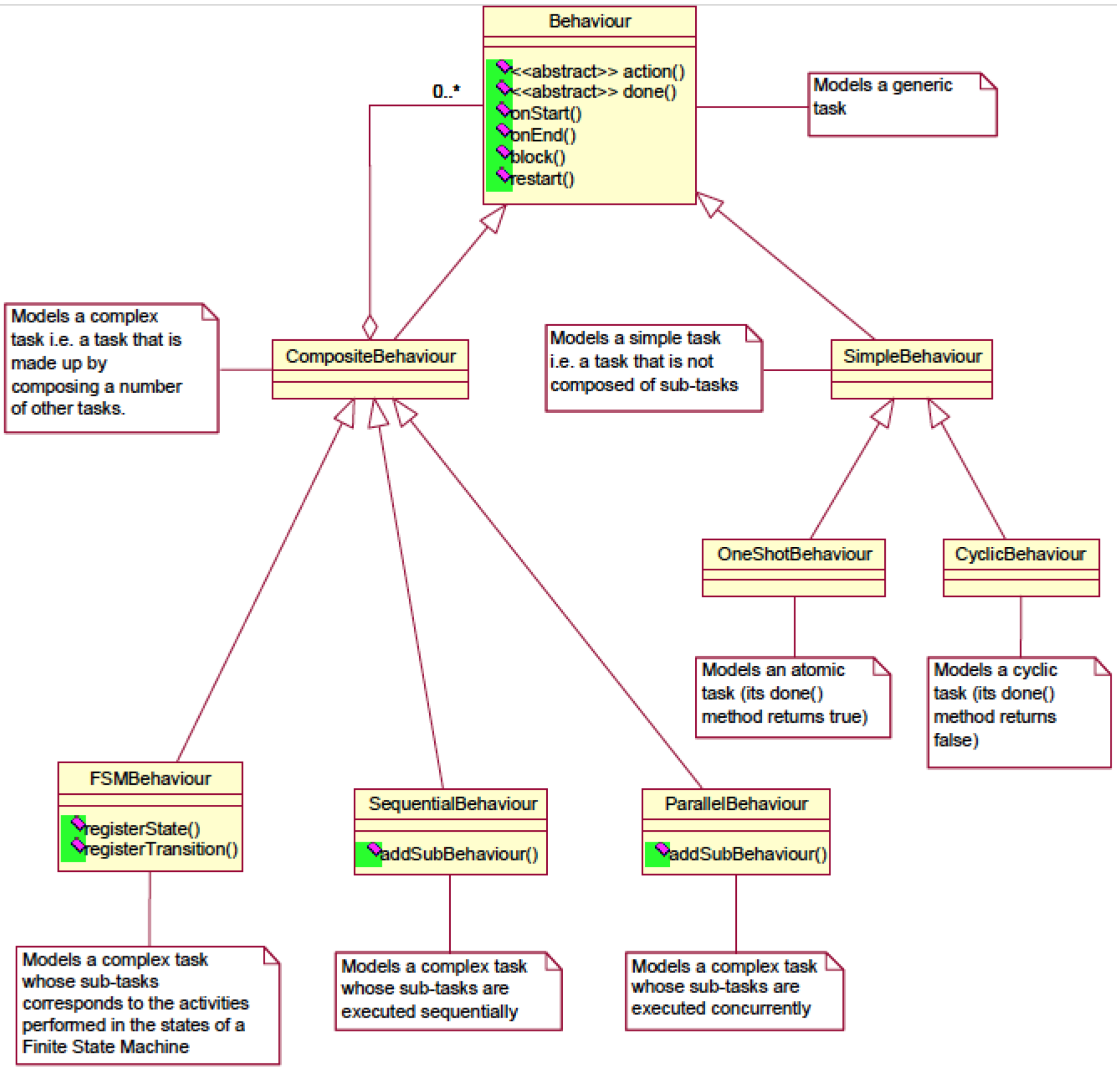

Behaviour
<<abstract>> action()
<<abstract>> done()
onStart()
onEnd()
block()
restart()
0..*
Models a generic task
Models a complex task i.e. a task that is made up by composing a number of other tasks.
CompositeBehaviour
Models a simple task i.e. a task that is not composed of sub-tasks
SimpleBehaviour
OneShotBehaviour
CyclicBehaviour
Models an atomic task (its done() method returns true)
Models a cyclic task (its done() method returns false)
FSMBehaviour
registerState()
registerTransition()
SequentialBehaviour
addSubBehaviour()
ParallelBehaviour
addSubBehaviour()
Models a complex task whose sub-tasks corresponds to the activities performed in the states of a Finite State Machine
Models a complex task whose sub-tasks are executed sequentially
Models a complex task whose sub-tasks are executed concurrently

Figure 4. An annotated UML class diagram for JADE behaviour

The above class hierarchy (Figure 4) is defined in the `jade.core.behaviours` package of the JADE framework. JADE differs the following primitive behaviours: **SimpleBehaviour**, **CyclicBehaviour**, and **OneShotBehaviour**. Composite behaviours in JADE are **ParallelBehaviour**, **SequentialBehaviour** and **FSMBehaviour**. The abstract class Behaviour supports (i) modelling of agent tasks, and (ii) behaviour scheduling (starting, blocking and restarting of a behaviour object). The `block()` method allows to block a behaviour object until certain event happens. A behaviour can be explicitly restarted by calling its `restart()` method. It also provides two methods, named `onStart()` and `onEnd()`. These methods can be overridden by user defined subclasses when some actions are to be executed before and after running behaviour execution. `onEnd()` returns an int (integer) that represents a termination value for the behaviour. The rest of this section further describes each of agent's behaviours supported by JADE (JADE Guide, 2010).

3.2.1 Primitive Behaviours

- Class **SimpleBehaviour**: This abstract class models simple atomic behaviours. Its `reset()` method can be overridden by user defined subclasses.

- Class **CyclicBehaviour**: This abstract class models atomic behaviours that must be executed forever. This behaviour stays active as long as its agent is alive and will be called repeatedly after every event. Quite useful to handle message reception. Its `done()` method always returns false.
- Class **OneShotBehaviour**: This abstract class models atomic behaviours that must be executed only once and cannot be blocked. Its `done()` method always returns true.

3.2.2 Composite Behaviours

- Class **CompositeBehaviour**: This abstract class models behaviours that are made up by composing a number of other behaviours (children). In particular this class provides a common interface for children scheduling, but does not define any scheduling policy. The scheduling policy must be defined by subclasses (SequentialBehaviour, ParallelBehaviour and/or FSMBehaviour).
- Class **SequentialBehaviour**: This class is a CompositeBehaviour that executes its sub-behaviours sequentially and terminates when all sub-behaviours are done. It is used when a complex task can be expressed as a sequence of atomic steps (e.g. do some computation, then receive a message, then do some other computation).
- Class **ParallelBehaviour**: This class is a CompositeBehaviour that executes its sub-behaviours concurrently and terminates when a particular condition on its sub-behaviours is met. Proper constants to be indicated in the constructor of this class are provided to create a ParallelBehaviour that ends when all its sub-behaviours are done, when any one among its sub-behaviour terminates or when a user defined number N of its sub-behaviours have finished. It is used when a complex task can be expressed as a collection of parallel alternative operations, with some kind of termination condition on the spawned subtasks. In other words, the important thing about ParallelBehaviour is the termination condition: we can specify that the group terminates when **ALL** children are done, **N** children are done, or **ANY** child is done.
- Class **FSMBehaviour**: This class is a CompositeBehaviour that executes its children (subclasses) according to a Finite State Machine (FSM), which is defined by the user. Each child (subclass) represents the activity to be performed within a state of the FSM and the user can define the transitions between the states of the FSM. When the child corresponding to state `Si` completes, its termination value (as returned by the `onEnd()` method) is used to select the transition to fire and a new state `Sj` is reached. At next round, the child corresponding to `Sj` will be executed. Some of the children of an FSMBehaviour can be registered as final states. The FSMBehaviour terminates after the completion of one of these children.
- class **WakerBehaviour**: This abstract class implements a one-shot task that must be executed only once, after a given timeout is elapsed.
- class **TickerBehaviour**: This abstract class implements a cyclic task that must be executed periodically.

JADE also provides other Behaviour such as SimpleAchieveREInitiator, and

SimpleAchieveREResponder.

3.2.3 Agent Behaviour in UNDERSTANDER

In case of agents in UNDERSTANDER, we're using the classes SequentialBehaviour, ParallelBehaviour, SimpleBehaviour, CyclicBehaviour, OneShotBehaviour and WakerBehaviour. The ParallelBehaviour is useful only when phases of parallel activity within more complex patterns such as Sequential or Cyclic activity is required. The CyclicBehaviour is active as long as its agent is alive and is useful to handle message reception. Figures 5-6 illustrates the way on which behaviours are invoked via the Client and the Server agents in UNDERSTANDER, respectively. For example, Figure 5 shows the Client agent invoking `WaitServerResponse` class, which extends ParallelBehaviour, to handle the task of sending message (message about contacting server). Class `ReceiveResponse` extends SimpleBehaviour and implements receiving of servers response, while a WakerBehaviour is added to terminate the waiting if there is no response from the server.

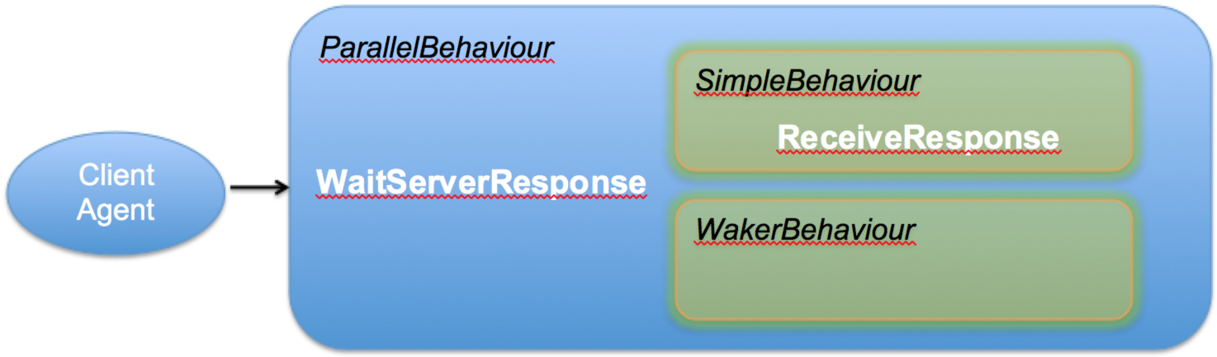


Figure 5. Client Agent in UNDERSTANDER and its "behaviours"

Figure 6 shows the Server agent's behaviour, which `setup()` method sets the agent's main behaviour, which is SequentialBehaviour. It invokes: `RegisterInDF()` method (OneShotBehaviour) and `ReceiveMessages()` method (CyclicBehaviour). The Directory Facilitator (DF) is a centralized registry of entries which associate service descriptions to agent IDs. The same basic data structure, the DFAgentDescription (DFD), is used both for *adding* an entry or *searching* for services. The difference is that ***when registering***, you provide a complete description and an AID; whereas ***when searching***, you provide a partial description with no AID. The search returns an array of complete entries (with AIDs) whose attributes match your description and you can extract the ID of suitable agents from those entries.

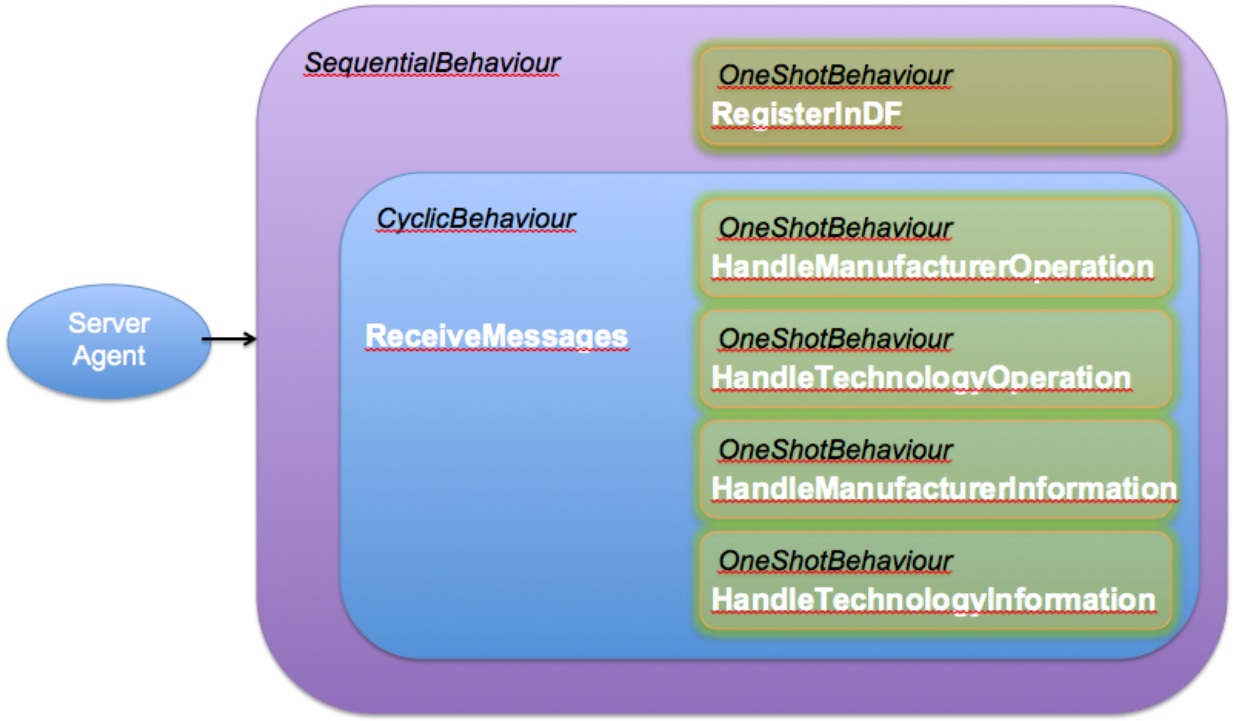


Figure 6. Server Agent in UNDERSTANDER and its "behaviours"

The following code additionally illustrates the Client agent's behaviour. Firstly, we import Java libraries needed to implement the ParallelBehaviour, SimpleBehaviour, and WakerBehaviour. We also import the knowledge base (ontologies) developed in D.4 "Business Intelligence Knowledge Base" (WP4) (i.e. HomeHeatingOntology and HomeHeatingVocabulary). Secondly, we define `sendMessage()` method that includes `WaitServerResponse()` method that extends ParallelBehaviour. Finally, `WaitServerResponse()` method includes `ReceiveResponse()` method (that is SimpleBehaviour) and WakerBehaviour that interrupts the programme after 5000 msec.

```
...
import jade.core.behaviours.ParallelBehaviour;
import jade.core.behaviours.SimpleBehaviour;
import jade.core.behaviours.WakerBehaviour;
...
import ontologies.HHManufacturer;
import ontologies.HHTechnology;
import ontologies.HomeHeatingOntology;
import ontologies.HomeHeatingVocabulary;
import ontologies.Problem;
import ontologies.SearchingManufacturersOperation;
import ontologies.SearchingTechnologiesOperation;
...
```

```
void sendMessage(int performative, AgentAction action) {
// utility method
      if (server == null) lookupServer();
      if (server == null) {
      alertGui("Unable to localize the server!");
      return;
      }
      ACLMessage msg = new ACLMessage(performative);
      msg.setLanguage(codec.getName());
      msg.setOntology(ontology.getName());
      try {
      getContentManager().fillContent(msg,      new      Action(server,
action));
      msg.addReceiver(server);
      send(msg);
      alertGui("Contacting server... Please wait!");
      addBehaviour(new WaitServerResponse(this));
      }
      catch (Exception ex) { ex.printStackTrace(); }
   }
...
class WaitServerResponse extends ParallelBehaviour {
// adding a SimpleBehaviour to receive servers response and
// a WakerBehaviour to terminate the waiting
      WaitServerResponse(Agent a) {
      super(a, 1);
      addSubBehaviour(new ReceiveResponse(myAgent));
      addSubBehaviour(new WakerBehaviour(myAgent, 5000) {
             protected void handleElapsedTimeout() {
                  alertGui("No   response   from   server.   Please,   try
later!");
                  resetStatusGui();
            }
      });
      }
   }

class ReceiveResponse extends SimpleBehaviour {
// Receive and handle server responses
      private boolean finished = false;
      ReceiveResponse(Agent a) {
      super(a);
      }
      public void action() {
```

```
     ACLMessage msg = receive(MessageTemplate.MatchSender(server));

     if (msg == null) { block(); return; }
     if (msg.getPerformative() == ACLMessage.NOT_UNDERSTOOD){
          alertGui("Response from server: NOT UNDERSTOOD");
     }
...
          catch (Exception e) { e.printStackTrace(); }
     }
     resetStatusGui();
     finished = true;
     }
     public boolean done() { return finished; }
     public int onEnd() { command = WAIT; return 0; }
   }
```

The following lines of code illustrates the Server agent's behaviour. Similarly to the Client agent, the first step of the Server agent imports behaviour-related libraries, and home heating knowledge base (i.e. HomeHeatingOntology and HomeHeatingVocabulary). Furthermore, we define `setup()` method that sets the main behaviour (SequentialBehaviour) invoking `RegisterInDF()` method and `ReceiveMessages()` method. As shown in Figure 6, `RegisterInDF()` method extends OneShotBehaviour, while `ReceiveMessages()` method extends CyclicBehaviour, by invoking several subclasses (OneShotBehaviour): (i) to perform searching operation about home heating manufacturers and/or home heating technologies (REQUEST as shown in Figure 2), and (ii) query specific information (QUERY_REF as shown in Figure 3). These subclasses are the following: class `HandleManufacturerOperation`, `HandleTechnologyOperation`, `HandleManufacturerInformation`, and `HandleTechnologyInformation`.

```
...
protected void setup() {
...
     // Set the main behaviour
     SequentialBehaviour sb = new SequentialBehaviour();
     sb.addSubBehaviour(new RegisterInDF(this));
     sb.addSubBehaviour(new ReceiveMessages(this));
     addBehaviour(sb);
   }
   class RegisterInDF extends OneShotBehaviour {
// Register in the DF
     RegisterInDF(Agent a) {
     super(a);
```

```
      }
      public void action() {
      ServiceDescription sd = new ServiceDescription();
      sd.setType(SERVER_AGENT);
      sd.setName(getName());
      sd.setOwnership("Violeta");
      DFAgentDescription dfd = new DFAgentDescription();
      dfd.setName(getAID());
      dfd.addServices(sd);
...

   class ReceiveMessages extends CyclicBehaviour {
// Receive requests and queries from client agent
      public ReceiveMessages(Agent a) {
      super(a);
      }
      public void action() {
      ACLMessage msg = receive();
      if (msg == null) { block(); return; }
      try {
           ContentElement                  content               =
getContentManager().extractContent(msg);
           Concept action = ((Action)content).getAction();
           switch (msg.getPerformative()) {
        case (ACLMessage.REQUEST):
System.out.println("Request from " + msg.getSender().getLocalName());
        if (action instanceof SearchingTechnologiesOperation)
        addBehaviour(new HandleTechnologyOperation(myAgent, msg));
        else if (action instanceof SearchingManufacturersOperation)
        addBehaviour(new HandleManufacturerOperation(myAgent, msg));
                 else replyNotUnderstood(msg);
                 break;

        case (ACLMessage.QUERY_REF):
System.out.println("Query from " + msg.getSender().getLocalName());
         if (action instanceof ManufacturerInformation)
     addBehaviour(new HandleManufacturerInformation(myAgent, msg));
     else if (action instanceof TechnologyInformation)
     addBehaviour(new HandleTechnologyInformation(myAgent, msg));
                 else replyNotUnderstood(msg);
                 break;
…
   }
   class HandleManufacturerOperation extends OneShotBehaviour {
// Handler for an Operation request
      private ACLMessage request;
      HandleManufacturerOperation(Agent a, ACLMessage request) {
      super(a);
      this.request = request;
      }
      public void action() {
      try {
            ContentElement                  content                  =
getContentManager().extractContent(request);
            SearchingManufacturersOperation            smo           =
(SearchingManufacturersOperation)((Action)content).getAction();
            Object obj = processManuOperation(smo);
            if (obj == null) replyNotUnderstood(request);
            else {
                  ACLMessage reply = request.createReply();
                  reply.setPerformative(ACLMessage.INFORM);
                  Result result = new Result((Action)content, obj);
                  getContentManager().fillContent(reply, result);
                  send(reply);
                  System.out.println("Operation   about   manufacturer
processed.");
            }
      }
      catch(Exception ex) { ex.printStackTrace(); }
      }
   }
...
```

4. Conclusion

This report discusses the design and development of agents in UNDERSTANDER. We particularly draw the reader attention to the definition of our agent communication protocol and their behaviour. This report fully relies on UNDERSTANDER knowledge base (ontologies) which is previously developed in WP4, and described in D.4 “Business Intelligence Knowledge Base”. Our user agents are developed in JADE, by consulting the online manuals (JADE, 2004), (JADE Guide, 2010). As one of the early results of this task, we refer on the paper “UNDERSTANDER Business Intelligence Seeker - User Agent” that is presented at the miproBIS (Business Intelligence Systems) conference in 2014 (Damjanovic & Behrendt, 2014).

The next step is done by WP2, described in D.2 “Conceptual Dependency Scripts for Business Intelligence”, in which we try to connect CD theory with the searching functionality of our user agents developed in WP3.

References

(Ahamo & Aljawaherry, 2012) A. Y. Ahamo, and M. A. Aljawaherry, “Constructing a collaborative multi-agents system tool for real-time system requirements” In International Journal of Computer Science (IJCSI), Vol. 9, No. 4, 2012.

(Arthur, 1994) W. Arthur. Inductive reasoning and bounded rationality. Complexity in Economic Theory, 84(2):406–411, 1994.

(Balin & Truszkowski, 2002) Bailin, S. C., Truszkowski, W. Ontology negotiation between intelligent information agents. The Knowledge Engineering Review, Vol.17(1), pp. 7-19. 2002.

(Burnstein et al., 2003) Burnstein, M., McDermott, D., Smith, D.R., Westfold, S.J., Derivation of glue code for agent interoperation. Autonomous Agents and Multi-Agent Systems, Vol.6(3):265-286, 2003.

(Chira, 2007) C. Chira, “A multi-agent approach to distributed computing,” Computational Intelligence Report No, vol. 42007, 2007.

(Chuan, 2011) Z. Chuan, "A software collaborative development environment based on intelligent agents", In proceedings of the 3rd International Workshop on Intelligent Systems and Applications (ISA), pp. 1-4, 2011.

(Claus & Boutilier, 1998) C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In Proceedings of National Conference on Artificial Intelligence AAAI/IAAI, pages 746–752, 1998.

(Damjanovic & Behrendt, 2014) V. Damjanovic, W. Behrendt, “UNDERSTANDER Business Intelligence Seeker - User Agent”, *In Proceeding of the 37th International Convention miproBIS (Business Intelligence Systems) 2014,* Opatija, Croatia, 28 May, 2014.

(Ding, Finin, & McGuinness, 2010) Ding, L., Shinavier, J., Finin, T., McGuinness, D.L.: owl:sameAs and Linked Data: An Empirical Study. In: Second Web Science Conference. Raleigh, North Carolina (2010).

(Doherty et al., 2005) Doherty, P., Szalas, A. Lukaszewicz, W. Approximative Query Techniques for Agents with Heterogeneous Ontologies and Perceptive Capabilities, In Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning, 2004.

(Ferber, 1999) Ferber, J., 1999. Multi-Agent Systems: An Introduction to Artificial Intelligence. 1999.

(Ficici & Pollack, 2000) Ficici, S. and Pollack, J. A game-theoretic approach to the simple coevolutionary algorithm. In Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI). Springer Verlag, 2000.

(Giri, 2011) K. Giri, “Role of ontology in Semantic web,” DESIDOC Journal of Library & Information Technology, Vol. 31, No. 2, 2011.

(Glance & Huberman, 1994) N. Glance and B. Huberman. The dynamics of social dilemmas. Scientific American, 270(3):76–81, March 1994.

(Gog & Gan, 2005) W. T. Goh, and J. W. P. Gan, "A dynamic multi-agent based framework for global supply chain", In Proceedings of ICSSSM '05. 2005 International Conference on Services Systems and Services Management, Vol 2., pp. 981-984, 2005.

(Greenwald et al., 2002) A. Greenwald, J. Farago, and K. Hall. Fair and efficient solutions to the Santa Fe bar problem. In Proceedings of the Grace Hopper Celebration of Women in Computing 2002, 2002.

(Himmelspach et al., 2003) Himmelspach, R., Williamson, R.E., Wasteneys, G.O., 2003. Cellulose microfibril alignment recovers from DCB-induced disruption despite microtubule disorganization. Plant J 36**:** 565–575

(Iordan et al., 2008) V. Iordan, A. Naaji, and A. Cicortas, “Deriving ontologies using multi-agent systems,” WSEAS Transactions on Computers, vol. 7, no. 6, pp. 814-826, 2008.

(Janowicz & Hitzler, 2013). K. Janowicz, and P. Hitzler. Thoughts on the Complex Relation Between Linked Data, Semantic Annotations, and Ontologies. In Proceedings of the 6th International workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR 2013), pp. 41-44, ISBN: 978-1-4503-2413-7.

(JADE, 2004) JADE online manual. Online available: http://www.iro.umontreal.ca/~vaucher/Agents/Jade/primer3.html

(JADE Guide, 2010) JADE’s Programmer’s Guide. last update 2014. Online available: http://jade.cselt.it/doc/programmersguide.pdf

(Jennings, 2000) N. R. Jennings, “On agent-based software engineering” Artificial Intelligence, vol. 117, no. 2, pp. 277-296, 2000.

(Jiao et al., 2006) J. Jiao, X. You, and A. Kumar, “An agent-based framework for collaborative negotiation in the global manufacturing supply chain network” Robotics and Computer-Integrated Manufacturing, vol. 22, no. 3, pp. 239-255, 2006.

(Kapetanakis & Kudenko, 2002) S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI02), 2002.

(Lauer & Riedmiller, 2000) M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000.

(Lee & Wang, 2009) C.-S. Lee, and M.-H. Wang, “Ontology-based computational intelligent multi-agent and its application to CMMI assessment” Applied Intelligence, Vol. 30, No. 3, pp. 203-219, 2009/06/01, 2009.

(Malucelli and Oliveira, 2006) Malucelli, A., Palzer, D., Oliveira, E. Ontology-based Services to help solving the heterogeneity problem in e-commerce negotiations. Journal of Electronic Commerce Research and Applications - Special Issue Electronic data engineering: the next frontier in e-commerce, Vol.5(3), Elsevier, 2006.

(Malucelli, 2006) A. Malucelli, 2006. Ontology-based Services for Agents Interoperability. PhD Thesis. University of Porto, 2006. Online available: http://goo.gl/HXhbJ4

(Manzoni, 2009) Manzoni, S., 2009. AACIMP 2009 Summer School lecture by Sara Manzoni. "Mathematical Modelling of Social Systems" course. Online: http://www.slideshare.net/ssakpi/interactions-in-multi-agent-systems

(Marivate et al., 2008) V. N. Marivate, G. Ssali, and T. Marwala, "An intelligent Multi-Agent recommender system for human capacity building", pp. 909-915, 2008.

(Mills & Stufflebeam, 2005) F. Mills, R. Stufflebeam. Introduction to Intelligent Agents. 2005. Online: http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php

(Monte-Alto et al., 2012) H. Monte-Alto, A. Biasão, L. Teixeira, and E. Huzita, "Multi-agent applications in a context-aware global software development environment distributed computing and artificial intelligence," Advances in Intelligent and Soft Computing, pp. 265-272: Springer Berlin / Heidelberg, 2012.

(Mundhe & Sen, 2000) M. Mundhe and S. Sen. Evolving agent societies that avoid social dilemmas. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), pages 809–816, Las Vegas, Nevada, USA, 10-12 2000. Morgan Kaufmann. ISBN 1-55860-708-0.

(Nikolov & Motta, 2010) Andriy Nikolov, V.U., Motta, E.: Data Linking: Capturing and Utilising Implicit Schema Level Relations. In: International Workshop on Linked Data on the Web. Raleigh, North Carolina (2010).

(Nunes et al., 2011) I. Nunes, C. P. Lucena, U. Kulesza, and C. Nunes, "On the development of multi-agent systems product lines: A domain engineering process" Agent-Oriented Software Engineering X, Lecture Notes in Computer Science, pp. 125-139: Springer Berlin Heidelberg, 2011.

(Nwana, 1996) H.S. Nwana, Software Agents: An Overview. *Knowledge Engineering Review*, Vol. 11, No 3, pp.1-40, September 1996. Cambridge University Press.

(Ören & Yilmaz, 2012) Ören, T. I. & Yilmaz, L. (2012). Synergies of simulation, agents, and systems engineering. Expert Systems with Applications 39(2012), pp. 81-88.

(Ören et al., 2007) Ören, T. I., Ghasem-Aghaee, N., & Yilmaz, L. (2007). An ontology-based dictionary of understanding as a basis for software agents with understanding abilities. In Proceedings of the spring simulation multiconference, Norfolk, VA, March 25–29, 2007 (pp. 19–27, ISBN: 1-56555-313-6).

(Ören et al., 2009) Ören, T. I., Yilmaz, L., Kazemifard, M., & Ghasem-Aghaee, N., (2009). Multiunderstanding: A basis for switchable understanding for agents. In Proceedings of the summer computer simulation conference on simulation series. Istanbul, Turkey, July 13–16, 2009 (Vol. 41(3), pp. 395–402). Dan Diego, CA: SCS.

(Ören, 2000) Ören, T. I. Opening paper. Understanding: A taxonomy and performance factors. In D. Thiel (Ed.), Proceedings of FOODSIM'2000, Nantes, France, June 26–27, 2000 (pp. 3–10). San Diego, CA., 2000.

(Pakdeetrakulwong & Wongthongtham, 2013) U. Pakdeetrakulwong & P. Wongthongtham: State of the Art of a Multi-Agent Based Recommender System for Active Software Engineering Ontology. International Journal of Digital Information and Wireless Communications (IJDIWC) 3(4): 29-42, 2013.

(Panait & Luke, 2005) Panait, L., Luke, S., Cooperative Multi-Agent Learning: The State of the Art. Autonomous Agents and Multi_Agent Systems. 11, 387-434, 2005.

(Panait & Wiegand, 2003) L. A. Panait, R. P. Wiegand, and S. Luke. Improving coevolutionary search for optimal multiagent behaviors. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), 2003. Online: http://goo.gl/BJiZUL

(Panait et al., 2004) L. Panait, R. P. Wiegand, and S. Luke. A visual demonstration of convergence properties of cooperative coevolution. In Parallel Problem Solving from Nature — PPSN-2004. Springer, 2004.

(Paydar & Kahani, 2011) S. Paydar, and M. Kahani, “An agent-based framework for automated testing of web-based systems” Journal of Software Engineering and Applications, 2011.

(Pinto et al., 1999) Pinto HS, Gómez-Pérez A, Martins JP (1999) Some Issues on Ontology Integration. In Proc. of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends. Vol. 18, pp 7-1 - 7-12. Stockholm, Sweden, 1999.

(Qingning et al., 2003) H. Qingning, Z. Hong, S. Greenwood, “A multi-agent software engineering environment for testing Web-based applications”, In Proceedings of the 27th Annual International Conference on Computer Software and Applications, pp. 1039--1045, 2003.

(Romero et al., 2008) M. Romero, A. Viscaino, and M. Piattini, "Towards the definition of a multi-agent simulation environment for education and training in global requirements elicitation", In Proceeding of the Conference on Human System Interactions, pp. 48-53, 2008.

(Steels, 1998) Steels, L. The origins of ontologies and communication conventions in multi-agent systems. Autonomous Agents and Multi-Agent Systems, Vol.1(2):169-194, 1998.

(Suguri et al., 2001) Suguri, H., Kodama, E., Miyazaki, M., Nunokawa, H., Noguchi, S. Implementation of FIPA Ontology Service. In Proceedings of the Workshop on Ontologies in Agent Systems, AAMAS, Montreal, Canada, 2001.

(Takahashi, 2008) Takahashi, T. 2008. RoboCup Rescue - Agent Based Disaster Simulation System: Challenges and Lessons Learnt. Agents, Simulation and Application. (Eds. A.M. Uhrmacher and D. Weyns), Taylor and Francis.

(Tolk et al., 2007) Tolk, A., Diallo, S.Y., Turnitsa, C.D., 2007. Applying the levels of conceptual interoperability model in supporting of integratability, interoperability and composability for system-of-systems engineering. Journal for Systemics, Cybernetics and Informatics, 5(5). 65-74.

(Tolk, 2006) Tolk, A. 2006. What comes after the Semantic Web: Pads implications for the Dynamic Web. 20th Workshop on Principles of Advanced and Distributed Simulation (PADS’06), pp. 55-62. IEEE Computer Society.

(Tzitzikas & Meghini, 2003) Tzitzikas, Y., Meghini, C. Ostensive automatic schema mapping for taxonomybased peer-to-peer systems. In Proceedings of the 7th International Workshop on Cooperative Information Agents, Helsinki, Finland, 2003.

(Uhrmacher, 2002) Uhrmacher, A.M., 2002. Simulation for agent-oriented software engineering. First International Conference on Grand Challenges, (Eds. W.H. Lunceford and E. Page), SCS, San Diego.

(van Diggelen et al., 2005) van Diggelen, J., Jan Beun, R., Dignum, F, van Eijk, R. M., Meyer, J-J. Optimal Communication Vocabularies and Heterogeneous Ontologies, In: van Eijk, R. M., Huget, M. P., Dignum, F. (eds.), Developments in Agent Communication, LNAI 3396, Springer Verlag, Berlin Heidelberg, pp. 76-90, 2005.

(van Eijk et al., 2001) van Eijk, R. M., Boer, F. S., van der Hoek, W., Meyer, J-J. Ch. On Dynamically Generated Ontology Translators in Agent Communication, International Journal of Intelligent Systems, Vol.16, pp.587-607, 2001.

(Wiegand, 2003) Wiegand, R.P. Analysis of Cooperative Coevolutionary Algorithms. PhD thesis, Department of Computer Science, George Mason University, 2003.

(Wiesman and Roos, 2004) Wiesman, F., Roos, N. Domain independent learning of ontology mappings, In: Jennings, N., Sierra, C., Sonenbergm, L., Tamble, M. (eds.), AAMAS, ACM Press, New York, USA, pp.846-853, 2004.

(Williams et al., 2003) Williams, A., Padmanabhan, A., Blake, M. B.Local Consensus Ontologies for B2B-Oriented Service Composition, In: Rosenschein, J., Sandholm, T., Wooldridge, M., Yokoo, M. (eds.), AAMAS, pp. 647-654. ACM Press, Melborne, 2003.

(Willmott, 2001) Willmott, S., Constantinescu, I., Calisti, M. Multilingual Agents: Ontologies, Languages and Abstractions, In Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, Montreal, Canada, 2001.

(Wolpert et al., 1999) D. H. Wolpert, K. R. Wheller, and K. Tumer. General principles of learning-based multi-agent systems. In O. Etzioni, J. P. M¨uller, and J. M. Bradshaw, editors, Proceedings of the Third International Conference on Autonomous Agents (Agents'99), pages 77–83, Seattle, WA, USA, 1999. ACM Press.

(Wooldridge & Jennings, 1995) M. Wooldridge and N.R. Jennings,1995. *Intelligent agents: theory and practice*. 10(2). Knowledge Engineering Review. pp. 115–152

(Wooldridge, 2002) M. Wooldridge. An Introduction to Multiagent Systems. John Wiley and Sons Ltd, February 2002.

(Yilmaz & Ören, 2009a) Yilmaz, L., & Ören, T. I. (Eds.). (2009a). Agent-directed simulation and systems engineering. Berlin, Germany: Wiley.

(Yilmaz & Ören, 2009b) Yilmaz, L., & Ören, T. I. (2009b). Agent-directed simulation (ADS). In L. Yilmaz & T. I. Ören (Eds.), Agent-directed simulation and systems engineering (pp. 111–143). Berlin, Germany: Wiley.

(Yilmaz and Tolk, 2006) Yilmaz, I. and Tolk, A., 2006. Engineering ab initio dynamic interoperability and composability via agent-mediated introspective simulation. Winter Simulation Conference, pp. 1075-1182.

(Zhong et al., 2004) Z. Zhong, J. D. McCalley, V. Vishwanathan, and V. Honavar, "Multi-agent system solutions for distributed computing, communications, and data integration needs in the power industry", In IEEE Power Engineering Society General Meeting, pp. 45-49 Vol.1, 2004.