PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 33/08

Using Reputations to Diagnose and Recommend Execution Plans to Software Agents in Ubiquitous Computing Systems

Andrew Diniz da Costa
Carlos José Pereira de Lucena
Viviane Torres da Silva
Donald D. Cowan
Paulo Sérgio C. de Alencar
Brian G. Keedwell

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 33/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena August, 2008

Using Reputations to Diagnose and Recommend Execution Plans to Software Agents in Ubiquitous Computing Systems

Andrew Diniz da Costa, Carlos José Pereira de Lucena,
Viviane Torres da Silva[1], Donald D. Cowan[2],
Paulo Sérgio C. de Alencar[2], Brian G. Keedwell[3]

[1] Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Madrid, Spain
[2] University of Waterloo, Waterloo, Canada
[3] Integrated Marketing and Inspiring Mobile Processes

{acosta, lucena}@inf.puc-rio.br, viviane@fdi.ucm.es,
dcowan@csg.uwaterloo.ca palencar@cs.uwaterloo.ca, brian.keedwell@telia.com

Abstract. Multi-Agent Systems (MAS) organized as collections of autonomous and heterogeneous agents working together to achieve a set of goals is a new paradigm in the software engineering of complex and distributed systems. However, it may be the case that the collection of agents is not able to attain their goals owing to failures during the execution of their related plan. When an agent tries to achieve its desired goals, but faces failures during execution, it becomes important to understand why such failures occurred and what can be done to remedy the problem. In this paper, we discuss solutions to the main challenges of performing diagnoses, namely determining what caused the failure of a plan, and to provide recommendations for alternate plans so that the collection of agents may repeat its attempt to achieve its goal. We also propose a hybrid diagnostic-recommendation framework that provides support for different methods of addressing such challenges. We focus on ubiquitous computing applications to demonstrate the proposed framework.

Keywords: Multi-Agent Systems, Diagnosis, Recommendation, Reputation, Ubiquitous Computing.

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Introduction 1

2 Diagnosing and Providing Recommendations 2

3 The DRP-MAS Framework 3

- 3.1 The General idea 4
- 3.2 Architecture 5
- 3.3 Diagnosis Module 6
  - 3.3.1 Data to be used while providing Diagnoses 6
  - 3.3.2 Strategies to provide Diagnoses 8
- 3.4 Recommendation Module 10
  - 3.4.1 Selecting Plans 11
  - 3.4.2 Verifying Selected Plans 11
  - 3.4.3 Choosing agents 12
- 3.5 Reputation Model 12
  - 3.5.1 Reputation provided by the Report framework 12
  - 3.5.2 Reputations provided by the Fire system 12

4 Case Study: Ubiquitous Computing with MPS 13

- 4.1 Translation 13
- 4.2 Scenario: Music Market Place 16

5 Related Work 17

6 Conclusions 18

References 18

1 Introduction

Multi-Agent Systems (MAS) [Jennings and Wooldridge, 1999] [Wooldridge and Ciancarini, 2000] organized as collections of autonomous and heterogeneous agents is a new paradigm for the software engineering of complex and distributed systems [Boella and Torre, 2004]. The agents are goal-oriented entities that may interact while executing their plans to try to achieve their goals. However, they may not able to attain their goals owing to failures during the execution of their plans. When an agent tries to achieve its desired goal but fails, it becomes important to understand why such failures occurred, namely to diagnose the problems, and to seek a solution to the problem by recommending other plans that will attempt to achieve the goal.

Interesting applications, which illustrate this idea, are present in the ubiquitous computing context. Ubiquitous computing assumes a world in which people are surrounded by mobile or fixed devices with a computing environment that supports them in almost all tasks. Since these services are typically provided by heterogeneous agents designed and implemented by different developers, it is reasonable to consider that failures may occur. Therefore, to diagnose the problem and provide associated recommendations to agents to attain their original goals is fundamental to an agent's successful execution.

Other approaches have been proposed in the literature to the problem of providing diagnosis and recommendations. Li et al. [2004] present a decentralized system to monitor and diagnose agents' behavior. In this system each agent or component, has an associated monitor to gather relevant data during the system execution. The information is then provided to a group of agents that work together to perform the diagnoses. Although this is an interesting idea, it is not applicable to open multi-agent systems, because monitoring appears to violate an agent's privacy.

In Horling et al. [2000], the authors examine how to apply domain independent diagnoses in multi-agent systems. This paper provides some very useful concepts related to diagnoses, which includes techniques to describe the correct behavior of an agent, and to compare this behavior with the actual result of execution. However, there is not a mechanism to discover if the failures were caused by the partners with whom the agent interacted. False or wrong information provided by a partner can influence an agent attaining its goals. Therefore, it is important to find out if the failure was provoked by the use of information from agent partners and to be able to select more trustful agents in future interactions.

In this paper we describe a new hybrid diagnostic-recommendation system to assist an agent in diagnosing its failures and to recommend alternative plans to support an agent in achieving its goals. A diagnostic system must be able to analyze different sets of information related to the agent's execution and to provide proper diagnosis by assembling all the facts and indicating the (main) problem that occurred. A failure may have many causes such as: the resource that an agent (provider of a service) is to use is not available or is damaged; an agent that normally collaborates in the execution of a given task chooses not to collaborate in a specific situation; or the information provided by an agent is inadequate.

Recommendations based on the diagnosis should provide alternative strategies to support an agent achieving the same goal. A recommendation system can recommend actions such as the use of another resource, the execution of another plan, or interacti-

on with other agents. While recommending other agents for interaction, the recommendation system bases its choice on the new partner's reputation. Reputation is a social notion associated with how trustworthy an individual is observed to be within a society [Koogan and Houaiss et al., 1995]. Several reputation models have been proposed to collect, aggregate and distribute feedback about an agent's past behavior.

The hybrid diagnostic-recommendation framework being proposed in this paper is called DRP-MAS (Diagnosing and Recommending Plans in open Multi-Agent Systems). The framework can be instantiated in different application domains and a variety of diagnoses and recommendations can be implemented. In Section 2 we discuss some of the main difficulties of diagnosis and providing alternative execution strategies for agents to achieve their goals in ubiquitous computing systems. In Section 3 we provide an overview of the DRP-MAS framework. Section 4 describes a complex problem associated with ubiquitous computing that is supported by our approach. Section 5 presents two simple case studies involving ubiquitous computing. Section 6 contains a description of some related work and Section 7 concludes and indicates possible future research directions.

2 Diagnosing and Providing Recommendations

In this section, we analyze the main requirements that were used in the design of the DRP-MAS framework. The main challenges are related to performing diagnoses and providing recommendations to achieve a goal that was not previously achieved by an agent. The challenges that we have identified are presented in the next few paragraphs.

1. Deciding how to analyze the behavior of agents

The first challenge focuses on the analysis of an agent's behavior. Two approaches have been considered. In the first one, the execution of each agent should be monitored. Since such an approach violates the agent's privacy, it was discarded. In the second approach, each agent is able to detect its own failures and to provide related information to be used in the diagnosis of the problem.

2. Selecting data for diagnosis

The second challenge is related to the selection of appropriate data to produce a diagnosis related to the execution of the failed agent. To produce a diagnosis, different information might be used, such as: the record of successful and failed communications with other agents, and the description of the problem arising from the availability of needed resources. A list of such data is identified and recorded.

3. Determining strategies for diagnoses

As different application areas use domain-specific information to make diagnoses, our challenge is to define a flexible approach that can be adapted to those different domains.

4. Determining trustworthy agents

While executing a plan, an agent may need to interact with other agents. When this interaction occurs, the information received by an agent can determine whether it will be successful in achieving its goal. Therefore, when a diagnosis is formulated and it is verified that a specific agent was involved in the unsuccessful execution, it may be appropriate to indicate that the reputation of that agent has been decreased in order to discourage future interaction. Reputation is a social notion associated with the amount of trust that can be placed on an individual's actions or the amount of faith one is willing to assign to someone else's integrity [Koogan and Houaiss, 1995].

5. Determining strategies for recommendations

As well as the strategies used to provide diagnoses, the strategies for generating recommendations may be domain-dependent, since domain-dependent data can determine the recommendation that matches the diagnosis. Therefore, our challenge is to provide a flexible approach that could be adapted to different domains and to provide a default basic strategy (based on the domain-independent information) that could be used in several domains.

6. Providing support to ubiquitous computing systems

Our approach offers a component that performs diagnoses and provides recommendations related to ubiquitous computing systems. In this context, we can encounter several pieces of important data that support diagnoses and recommendations, such as different devices and types of connections used. Different devices may affect a request made by an agent to a service. For example, an agent may have to send a dataset in a specific format to a device in order to ensure correct reception and interpretation. The challenge is to organize the limitations involving different devices as this may affect both diagnosis and recommendations. Different types of connections in the ubiquitous computing domain are very common as people access the Internet from different locations at different speeds. The challenge is to determine which data-connection pairs can be used to perform effective diagnoses and provide recommendations.

7. Representing profiles of agents

Each agent has particular properties and characteristics that can be represented in a profile. Our challenge is to define a component that could be used to represent generic agent profiles and to provide services that help diagnosis and recommendation strategies use these profiles.

3 The DRP-MAS Framework

In this section, we present the DRP-MAS framework (Diagnosing and Recommending Plans in Multi-Agent Systems). The framework assists agents in achieving their goals by providing the infrastructure to support different diagnoses and provide recommendations.

3.1 The General idea

The DRP-MAS framework is used when an application agent does not achieve one of its goals after the execution of one of its plans. Such an application agent, called *Requester agent*, sends a request to *Mediator agent* that creates a specific *Diagnostic agent* (responsible for providing diagnoses) and a *Recommendation agent* (responsible for providing recommendations). After that, the *Mediator agent* sends a message to the *Requester agent* indicating the *Diagnostic agent* with which to interact (**Error! Reference source not found.**).

Once the specific *Diagnostic agent* is created and identified, the *Requester* requests advice from the *Diagnostic agent* in order to achieve its desired goal (Figure 2). For that purpose, it sends a message to the *Diagnostic agent* with the set of information that can help it in the analysis, such as: plan executed, goal not achieved, the agents with whom it has interacted, its profile, and a number that represents the quality of the service performed [Horling et al., 2000], [Wagner et al., 2003] and [Lesser et al., 2007]; more details are provided in Section 3.3.

When the *Diagnostic agent* receives the message, it tries to discover what has caused the failure in a *Requester agent* execution. At the end of the analysis, the Diagnostic *agent* provides the diagnosis to the *Recommendation agent*, which then provides a recommendation. Even if a diagnosis could not be provided, the *Diagnostic agent* sends a message to the *Recommendation agent* stating that it was not possible to determine why the *Requester agent* did not achieve the desired goal. In this case, when the *Recommendation agent* receives the message indicating it was not possible to provide a diagnosis; it still tries to select plans that can be used to achieve the desired goal.

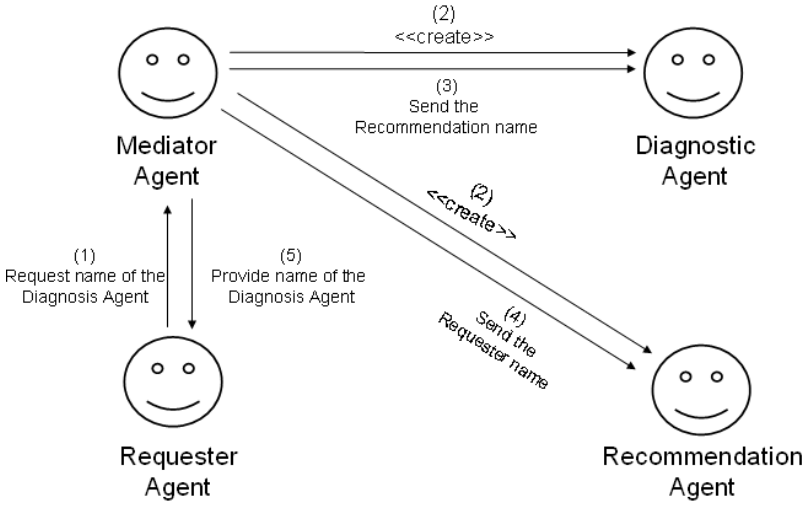

Figure 1. Conceptual Model for requesting the name of the diagnostic agent

In the case in which a diagnosis is provided, the *Recommendation agent* searches for alternative plans to achieve the goal (details shown in Section 3.4) using the diagnosis information. When the diagnosis indicates a problem in the interaction with a specific agent, an analysis is made to decide which other agents can be used to perform the interactions or services identified in the recommended plans.

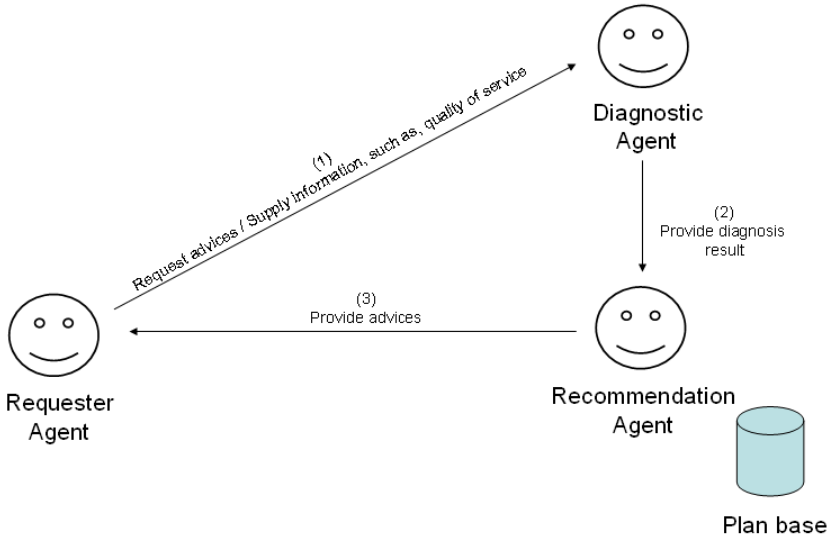

Figure 2. Conceptual Model for requesting advice

From the set of agents that can perform the same services, the *Recommendation agent* selects the agents with highest reputations. The profile of the *Requester agent* can also be consulted while selecting the agents. When the *Recommendation agent* finishes, a message is sent to the *Requester agent* with the recommendations, namely the alternative plans to achieve the same goal.

In order to be able to provide recommendations, the *Recommendation agent* must receive the list of plans the *Requester agent* can execute and their related information, such as: the goals of each plan, the services requested by the plan, and the resources used.

3.2 Architecture

In this section we describe the architecture of the DRP-MAS. As illustrated in Figure 3 the DRP-MAS sub-system communicates with the Application sub-system. The DRP-MAS is composed of five modules: Mediation, Diagnosis, Recommendation, Artificial Intelligence Toolset and Reputation.

The *Mediation* module is responsible for providing the *Mediator* agent (see section 3.1), which creates a *Diagnostic agent* and a *Recommendation agent* to interact with the *Requester agent* executing in the Application sub-system. The *Diagnosis* module performs the process of diagnosis, while the *Recommendation* module tries to provide recommendations to achieve some desired goal. The *Artificial Intelligence Toolset* module, which has an API (Application Programming Interface) called BIGUS [Bigus and Bigus, 2001], provides different reasoning algorithms (forward chaining, backward chaining and fuzzy logic) that could be used by the diagnosis and recommendation modules. The two most important modules, diagnosis and recommendation, will be detailed in Sections 3.3 and 3.4, respectively.

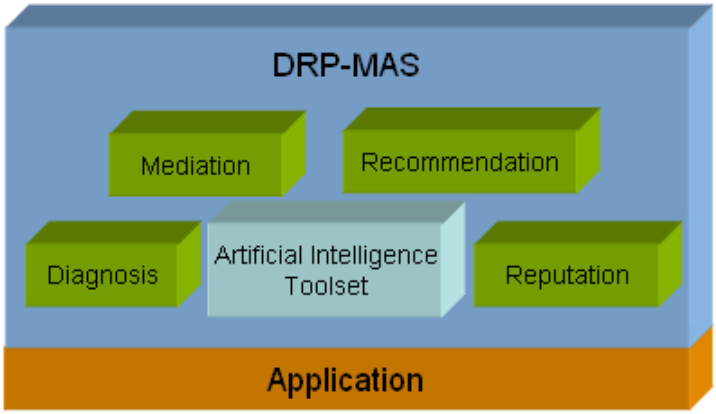

Figure 3. Architecture

The Reputation module provides reputations of other agents to the DRP-MAS and to the Application sub-system. In the current implementation, the reputation model is based on the Fire model [Huynh et al., 2004] and on the Report reputation system [Guedes et al., 2006] [Silva et al., 2007]. More details are in sub-section 3.5.

The DRP-MAS framework was implemented by using Jadex [Poukahr and Braubach, 2007] [Braubach et al., 2003], which is one of the most useful platforms for creating agents. We decided to choose this platform because it provides an implementation to the Belief-Desire-Intention (BDI) abstract architecture [Braubach et al., 2004] that is the basis for our diagnosis and recommendation approach. Our approach is based on the failures that occur while agents are trying to achieve their goals (or desires) through the execution of their plans (or intentions).

3.3 Diagnosis Module

As mentioned in section 3.1, the diagnoses are provided by the *Diagnostic agent* available in the framework. Such diagnoses are based on data provided by the *Requester agent*. Although different data can be required in different application domains, it is possible to state a set of domain-independent data that can be used while supplying the diagnoses. The data provided in section 3.3.1 and summarized in Table 1 can be used to provide both diagnoses and recommendations. In section 3.3.2 we describe the mechanisms available in the framework to help provide diagnoses together with an example of a strategy used to provide diagnoses.

3.3.1 Data to be used while providing Diagnoses

The following list presents the domain-independent data that the *Requester agent* should send to the *Diagnostic agent* in order to help with the generation of diagnoses. This data should also be supplied to the *Recommendation agent* to support the selection of alternative plans and partners.

1. Resources and associated problems – In [Horling et al., 2000] resources are considered important data to support diagnoses. A possible reason for an execution failure can be an insufficient amount or the absence of a resource.
2. Quality of service – As defined by the TAEMS model [Horling et al., 2000] [Wagner et al., 2003] [Lesser et al., 2007], a quantitative value can be used to evaluate a task

executed by an agent. Such a value can be used in the diagnoses because it indicates the success of the agent in performing the task.

3. Goal – The execution of an agent's plan is always associated with a goal that the agent wants to achieve [Silva et al. , 1999]. To know this goal is fundamental to both a diagnosis and the corresponding advice since the diagnosis will depend on the goal the agent wants to achieve and the alternative plans will be provided based on achieving the same goal.
4. Plan executed – In order to understand the reason for the failure and to provide alternative execution strategies it is necessary to have the plan that was executed by the *Requester agent*.
5. Agents with whom the agent interacted – It may be the case that an agent fails to achieve a goal because of wrong or false information received from another agent. Thus, it is important to know which agents were involved during the execution of the *Requester agent* plan.
6. Services used – It may be the case that a failure is caused by the poor quality of a service provided by another agent. Therefore, it is important to know the services used in order to provide diagnoses and to recommend other agents to provide the same services. Each request performed by some agent can have important identifying data: (i) request identifier, (ii) date and hour of the request and (iii) grade of severity of the request.
7. Profile – Each agent has a profile, which can be used to represent some of its characteristics. The *Requester agent* profile can, for instance, stipulate the minimum reputation value that its partners must have. Thus, the *Recommendation agent* should consider such information while seeking partners.
8. Belief Base – The knowledge base used by the *Requester agent*, including the time of its last update, can be useful to perform diagnoses and to provide recommendations.

In order to illustrate the set of data sent from the *Requester agent* to the *Diagnostic agent*, let's focus on the simple domain of making coffee. An agent has a goal to make coffee for friends and, therefore, executes a plan to achieve such a goal. Suppose that the agent notices that the coffee is not good but does not understand why. There are several reasons for producing bad coffee such as: the quality of the coffee powder is poor, the water used was cold, or the quantities of coffee and water were not adequate. To find out what has happened, the *Requester agent* should send to the *Diagnostic agent* information about its goal (to make coffee for three persons), the plan it has executed to make the coffee, the quality of the service representing how quality of the coffee, and other domain-dependent information such as the quantity of water, the temperature of the water and the quality and the description of the coffee powder.

In the case of ubiquitous computing applications, the following information, already identified in DRP-MAS, should also be provided by the *Requester agent*:

1. Device used – Because of the different characteristics of available devices, it is important to identify (i) the type of device used (ex: cell phone, PDA, laptop, etc), (ii) its model (ex: LG MG296 GSM, Motorola Razr V3 Black GSM, etc.) and (iii) the language in which the data must be provided by the agent (ex: English, Spanish, Portuguese, etc).
2. Connections – The characteristics of the connections, i.e., (i) its speed (ex: 56Kbps, 512Kbps, 3Mbps, etc), (ii) its technology (ex: wireless, LAN, WAN, etc.) and (iii) the

IP address used, are important to construct a diagnosis and to provide recommendations.

Table 1. Data about the quality of services

Data	Brief Description
Resources	Resources used during execution.
Quality of Execution	Number which defines the quality of execution of a plan. This idea is taken from the TAEMS model.
Goal	Goal not achieved.
Plan executed	Plan executed by the *Requester* agent that has not achieved the desired goal.
Agents negotiated	Name of agents whose execution requested information.
Service	Services used by the plan with the description of some request performed (ex: date, hour, identifier, severity of the request).
Profile	Profile of the *Requester* agent
Belief Base	Knowledge base of the agent.
Devices	*Description of the device used by a customer.*
Connections	*Data that describe the settings of the connection used, such as, speed and IP* address.

3.3.2 Strategies to provide Diagnoses

Since many different types of diagnoses can be performed by using domain-independent and domain-dependent data, the DRP-MAS framework has hot spots (flexible points) to support definition of diagnosis strategies. Note that different strategies can also be used by the same *Requester* agent depending on the data available after execution of the plan.

To help with the implementation of domain-dependent strategies three different algorithms (backward chaining, forward chaining and fuzzy logic) are available in the Artificial Intelligence Toolset module defined in the framework, as mentioned in subsection 3.2. The framework offers special support to the use of the forward chaining algorithm. The service provided by the framework helps with the identification of more precise diagnoses even when the user has provided very little data. In order to employ such a service the user of the framework must supply a hierarchy rule base – such as the one illustrated in Figure 4 – that will be used by the framework to discover related diagnoses.

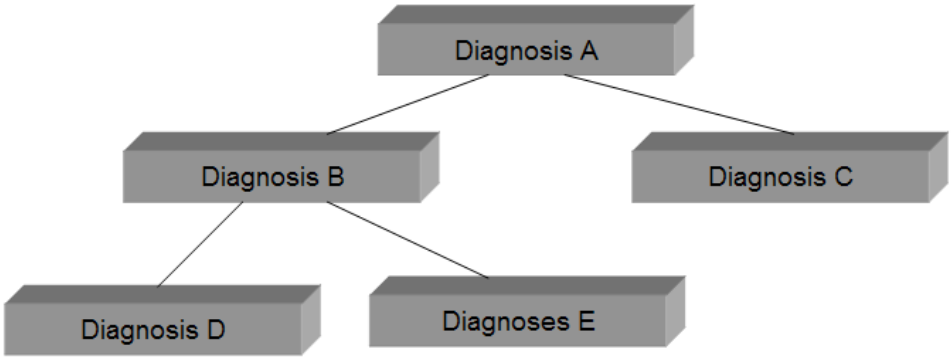

Figure 4. Relation of diagnoses

Let's suppose that the diagnosis B was determined by using the data provided by the *Requester agent*. Following the diagnoses hierarchy illustrated in Figure 4, it is possible to verify that two other more specialized diagnoses (D and E) could have been discovered if more information had been provided.

The service implemented in our framework is able to provide the *Recommendation agent* not only with the diagnosis determined by using the data provided by the *Requester agent* but also the other more specific diagnoses that could have been determined if more information had been supplied. Therefore, the *Recommendation agent* will be able to avoid selecting plans related to both the more general diagnosis as well as the more specific ones.

To illustrate the use of this service, let's suppose that a *Requester agent* wants to have a word translated from Portuguese to English and uses the service of a *translator agent*. When the *translator* does not successfully perform its task, the *Requester agent* asks for recommendations since it has not achieved its goals. When the *Diagnostic agent* receives the request, it uses the forward chaining algorithm to infer the diagnosis from the data provided by the *Requester*. Let's suppose that the *Requester* has provided the quality of the service (that is equal to 0 since no translation occurred), the word to be translated and the dictionary used. After receiving this information the *Diagnostic agent* checks if the word is in the dictionary and uses its rule base, shown in Table 2, to infer the diagnosis. Suppose that the word was not found in the dictionary and the diagnosis inferred was Error_Translation, i.e., unspecified error has occurred while trying to translate.

If the *Requester agent* had provided more information, two other diagnoses could have been provided, as illustrated in Figure 5. It may be the case that the word was not found because it was not correctly typed or because an old version of the dictionary is being used. Thus, DRP-MAS sent the Recommendation agent not only the real diagnosis but also the other two that could have been used if more information was provided.

Table 2. Rule Base of translation English to Portuguese

IF Quality_of_Service=0 **AND**
Base_Belief=FALSE **THEN**
DIAGNOSIS=ERROR_TRANSLATION
IF Quality_of_Service=0 **AND**
Base_Belief =FALSE **AND**
Base_Belief_Updated=FALSE **THEN**
DIAGNOSIS=NECESSARY_REQUEST_WORD
IF Quality_of_Service=0 **AND**
Base_Belief =FALSE **AND**
Base_Belief_Updated=TRUE **THEN**
DIAGNOSIS=NOT_POSSIBLE_TO_TRANSLATE

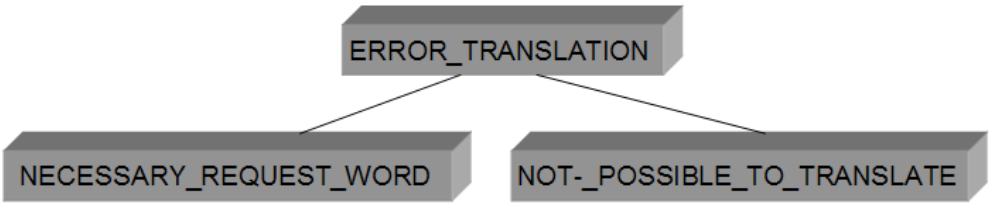

Figure 5. Relation between diagnoses defined on the rule base of translation

3.4 Recommendation Module

The *Recommendation agent* includes a process for providing alternative ways to achieve a goal. This process is composed of three steps (Figure 6):

1. Selecting plans,
2. Verifying whether the selected plans presuppose interactions with other agents,
3. Choosing appropriate agents.

The first step is executed when the *Recommendation agent* receives the diagnosis from the *Diagnostic agent* and needs to verify which plans can be used to achieve a specific goal. Based on the information provided by the *Requester agent*, the *Recommendation agent* analyzes the plans to be chosen.

The second step verifies if the selected plan implies interaction with other agents. If it is not necessary, then the process is finished and a message containing the recommended plans is sent; otherwise, the third step is executed. In the third step, candidate agents are selected using the *Reputation module* detailed in section 3.5. At the end, the selected plans and agents are provided to the *Requester agent*. The following subsections provide a further explanation of each step of this process.

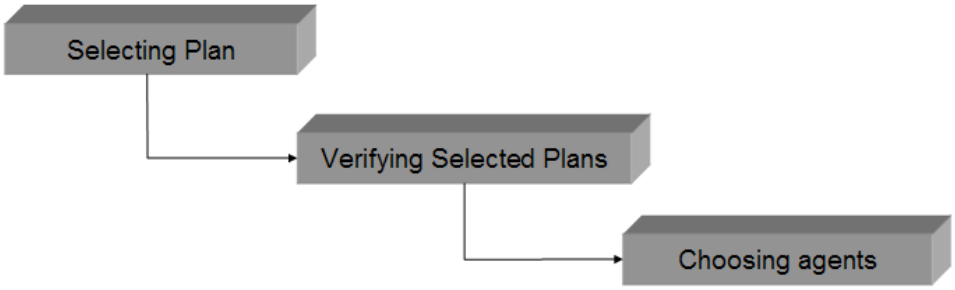

Figure 6. Execution process of the Recommendation agent

3.4.1 Selecting Plans

This step is responsible for choosing alternative plans to achieve the desired goal. Since each application may need to implement a different strategy, the DRP-MAS framework defines such a strategy as a hot-spot that can be instantiated based on the requirements of the application.

In order to be able to select plans based on the information provided by the *Requester agent*, each plan should be associated with a set of information that describes and classifies the plan. It is important to point out the services and resources used during the execution, the goal that the plan will try to achieve, related diagnoses, a value that specifies when the plan must be executed, and other domain-dependent information such as devices that the plan will use, types of associated connections, and a collection of possible problems that the plan can solve. Note that the information associated with each plan is strongly related with the information that the *Requester agent* can provide to the *Diagnostic agent*, as described in sub-section 3.2.

DRP-MAS offers two simple services with the aim of helping with the definition of the recommendation strategy. One of the services is able to select plans that are related to given data and the other is able to select plans that are not related to such data. After selecting the recommended plans, the second step of the recommendation process should be executed. In the case where no plan could be recommended, a message is sent to the *Requester* and the process is aborted.

3.4.2 Verifying Selected Plans

After the selection of alternative plans, this step verifies whether the plans will require the use of services that will be provided by other agents. If no plan requires such services, a message is sent to the *Requester agent* with the recommendations that have been obtained. Otherwise, the *Recommendation agent* will need to discover the agents that are able to provide such services and to select the ones with the best reputation. To select these trustworthy agents the *Recommendation agent* requests from the *Reputation* agent (detailed in Section 3.5) information about the reputation of the agents that can provide the required services.

3.4.3 Choosing agents

This step is responsible for filtering the more trustworthy agents from the set of agents that can provide the services. The *Recommendation agent* can also consider the *Requester* profile while selecting its partners. The strategy used to select the partners is the third important hot-spot of the framework. Since a different reputation model can be used and different profiles can be defined, DRP-MAS defines the strategy used to chose the partner as a flexible point.

3.5 Reputation Model

The reputation model is the one responsible for providing the reputations of agents that provide a given service. Although the framework offers a default implementation of the *Reputation agent* based on the reputations provided by the Report framework [Guedes et al., 2006] [Silva et al., 2007] and the Fire system [Huynh et al., 2004], any other reputation system can be used. The strategy used by the *Reputation agent* to select the reputations of the agents is a hot-spot of the framework. The default strategy provided by the framework uses three different types of reputation provided by the Fire model and one provided by the Report framework to evaluate the reputations of the agents.

3.5.1 Reputation provided by the Report framework

The Report framework implements a centralized reputation mechanism that stores the reputations of the application agents. The agents in the system are able to provided information about the behavior of other agents and Report, as part of the Governance framework, is able to evaluate such behavior and store the associated reputations.

Based on the centralized reputation mechanism defined in Report, DRP-MAS defines a reputation base to store reputations provided by agents about the behavior of others. After interacting with their partners, the agents should evaluate their behavior and send such information to DRP-MAS. The main idea is to define a unique reputation (or global reputation) for each participant in an application, and to allow agents access to such global reputations.

3.5.2 Reputations provided by the Fire system

Fire defines a decentralized reputation mechanism where the agents are able to evaluate the behavior of other agents and also to store their reputations. From the set of available trust and reputations types defined in Fire, DRP-MAS uses the following:

- interaction trust (resulting from past experiences of direct interactions);
- witness reputation (reports of witnesses about other agents' behavior);
- certified reputation (references provided by other agents about an agent's behavior).

Fire was chosen because it defines certified reputations which is an extra category not provided by other reputation systems. Such reputations are fundamental when an

agent wants to know the reputation of agents with which it has not interacted and when it does not know any other agent that has interacted with the desired agent.

By using the *Reputation agent*, the *Recommendation agent* is then able to ask for agents exhibiting any of the four available reputation types defined as default in DRP-MAS. As stated before, other reputation systems and associated reputation types can be implemented.

4 Case Study: Ubiquitous Computing with MPS

The two case studies presented in this section are related to the domain of Mobile Process Service (MPS), a business process that brings together individuals solving a problem by sharing expertise while interacting through communication devices driven by agents. Since such agents are typically heterogeneous, potentially designed and developed by different developers and also distributed over the environment, it is reasonable to suggest that failures may occur. Therefore, this domain is appropriate to illustrate our diagnosis and recommendation framework. Our approach provides a diagnosis when an agent cannot complete its assigned task and alternative plans.

The two simple scenarios used to illustrate our approach are: the translation of a word and the negotiation of the price of a music CD in a market place.

4.1 Translation

In this sub-section, we explain the translation scenario and the corresponding implementation we have developed using the DRP-MAS framework. In this setting a customer needs to have a word translated from Portuguese to English. This customer uses a service provided by a *Translator agent*. Depending on the characteristics of the device used by the customer (a laptop or a cell phone), different information is provided by the *Translator.* If the device is a cell phone, the *Translator* sends only the word translated to English. However, if the device is a laptop, a link to a ".txt" document is provided, which contains: (i) the word requested, (ii) the word translated, and (iii) the meaning of the word in Portuguese and in English.

When the *Translator* agent receives a request, it tries to perform the translation using the dictionary stored in its belief base. If the word is found in the belief base, the agent performs the translation and provides the result to the customer. Otherwise, it uses DRP-MAS to receive recommendations of alternative plans in order to translate the word.

The first step executed by DRP-MAS diagnoses the execution performed by the *Translator* based on the set of information it provides: the device used by the customer (cellphone or laptop), its belief base, the desired goal (to perform the translation), the plan executed, the quality of the plan executed, the last time that the belief base was updated, and the word to be translated.

The strategy to provide diagnoses in this scenario uses the forward chaining algorithm. We defined four possible diagnoses as illustrated in Table 3 and Figure 7: (i) the word was not correctly typed, (ii) the *Translator* does not know the word requested, (iii) the word does not exist and (iv) the belief base is outdated.

The diagnosis stating that the word was not correctly typed is determined when the quality of execution is lower than ten (ten represents success on the translation), and when the *Translator* finds words very similar to the word provided by the customer, i.e., words that only differ in one or two letters. In such a case, we are assuming that the customer omitted one or two letters, or typed the wrong letter. Thus, the *Recommendation agent* recommends plans that try to find similar words and provide the translation of such similar words when the word is not directly found in the belief base.

The second possible diagnosis, "*Translator does not know the word requested*", is determined when the quality of execution is lower than ten and the *Translator* has not found any similar word. Two other more specific diagnoses can be found if the *Translator* is able to provide more information about the execution (Figure 7). If the *Translator* agent provides to the *Diagnostic* agent the information about the last time its belief base was updated, a more specific diagnosis can be met. It is possible to conclude that "*Belief base is outdated*" because it is using an old dictionary. In such case, the *Recommendation agent* will propose plans that, before trying to translate the word, ask other *Translators* for more up-to-date dictionaries.

Table 3. Rule Base of the translation scenario

IF Quality_Execution < 10 AND
SimilarWord = TRUE
THEN **Diagnosis = "Word incorrectly typed"**
IF Quality_Execution < 10 AND
SimilarWord = false
THEN **Diagnosis = "Translator does not know Word"**
IF Quality_Execution < 10 AND
SimilarWord = false AND
Updated = true
THEN **Diagnosis = "Word does not exist"**
IF Quality_Execution < 10 AND
SimilarWord = false AND
Updated = false
THEN **Diagnosis = "Belief base outdated"**

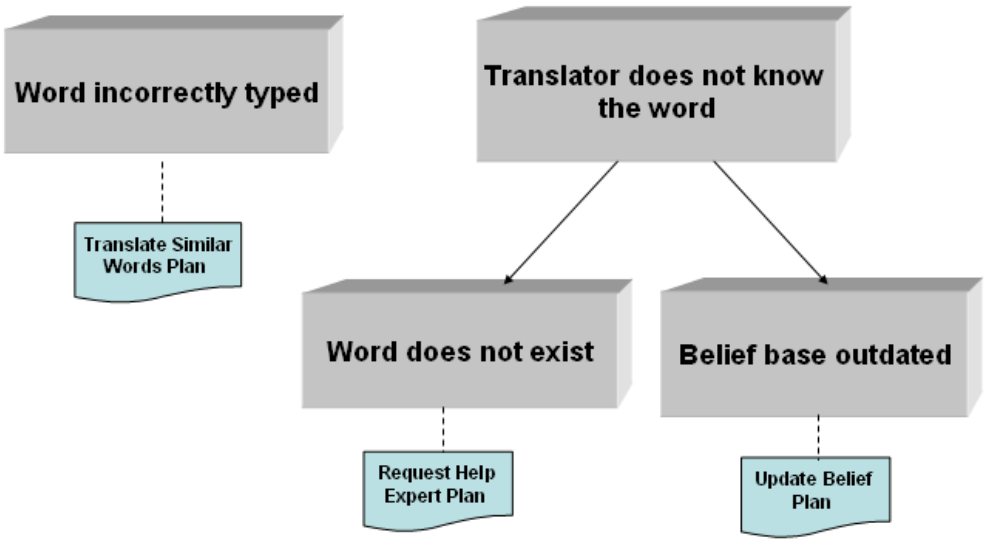

Figure 7. Relation between the diagnoses and their respective plans

If the *Diagnostic agent* concludes that the belief base is up-to-date, the diagnosis provided is "*Word does not exist.*" The plan recommended by the *Recommendation agent* is the one that asks a person to translate the word.

When it is needed to ask for up-to-date dictionaries from other *Translators* the *Recommendation agent* selects the possible translators based on their reputations and on the profile of the translator asking for help. It interacts with the *Reputation agent*, and in this example, asks for the global reputations of the translators.

Figure 8 illustrates the use of DRP-MAS by a *Translator agent (TA)*. In the scenario illustrated in the Figure, the alternative plan executed by the *TA* forces it to interact with other *TAs* in order to update its belief base.

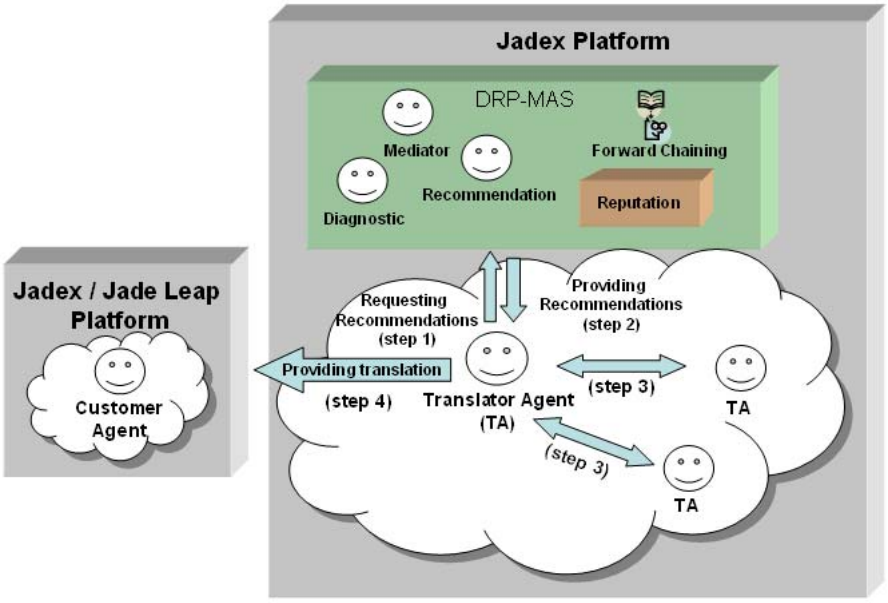

Figure 8. Translator agent updating belief base

4.2 Scenario: Music Market Place

In this section, we explain the music market place scenario, which was implemented using the DRP-MAS framework. In this scenario a customer or *Buyer agent* that wishes to buy CDs contacts a *Seller agent* and provides the name of the music that must be on the CD, its category and the maximum price the *Buyer agent* or customer is willing to pay for that CD.

If the *Buyer agent* does not receive the desired CD, it requests recommendations from DRP-MAS by providing (i) the plan executed; (ii) the quality of such execution; (iii) the desired goal ("to buy a specific CD"); (iv) the identification of the *Seller agent*; (v) the agent profile that defines the amount of money to be spent buying CDs and the minimum acceptable reputation of each of its partners; (vi) the name of the music; (vii) the chosen CD category and (viii) the CDs provided by the seller, if any.

To perform a diagnosis, we used the forward chaining algorithm again. A rule base (see Table 4) was defined with three possible diagnoses: (i) seller does not know of any CD with the characteristics specified by the buyer, (ii) the CD provided is more expensive then the amount the buyer is willing to pay, and (iii) although the CD has the specified music it belongs to a different category.

The first diagnosis occurs when the buyer has not received any CD since there is no CD with the desired music (quality_execution = 0 indicating that no CD was provided). The second diagnosis occurs when the CD provided by the seller is more expensive then the buyer is willing to pay (quality of execution between zero and ten indicating that a CD was provided but it is not the one the buyer wants). The third diagnosis happens when the CD provided has the desired music but is in a different category. We can see that, in this scenario, the diagnoses are totally independent (Figure 9), unlike the translation described in the previous section. On the other hand, in this scenario it is possible to have two diagnoses occur at the same time.

After receiving the diagnosis, the *Recommendation agent* updates the reputation of the seller according to the product it has sent to the buyer. If the seller has not provided any CD, its reputation decreases more than in the cases where a CD was provided.

Table 4. Rule base of the Music Marke Place scenario

IF Quality_Execution = 0 AND
THEN **Diagnosis = "Seller_Does_Not_Know_CD"**
IF Quality_Execution < 10 AND
Quality_Execution > 0 AND
Mash_profile = "false" AND
THEN **Diagnosis = "High_Price_CDs"**
IF Quality_Execution < 10 AND
Quality_Execution > 0 AND
THEN **Diagnosis = "Different_Category_of_Music"**

Figure 9. Diagnoses of the Music Market Place scenario

Since the three different diagnoses blame the seller for not providing the buyer with the desired CD, we can conclude that the plan the buyer has executed is not the problem. Therefore, the *Recommendation agent* recommends the same plan but suggests different sellers. The sellers are selected according to the buyer profile. In the case the buyer has not specified any maximum price in its profile, the *Recommendation agent* recommends expert agents, i.e., agents that will probably send more expensive CDs owing to their expertise on finding what their clients want. In the case where the buyer has specified a maximum price, the *Recommendation agent* recommends common sellers that have lower, but reasonable reputations.

5 Related Work

In Li et al. [2004], a decentralized system is proposed to perform diagnosis and monitoring. Each component has a monitor (Monitoring Agent), which is responsible for collecting information about the component. Once it has obtained the information, it is provided to agents responsible for finding diagnoses while working together. When applied to open multi-agent systems this approach violates the agents' privacy. DRP-MAS, on the other hand, avoids the creation of monitors and lets the application agents themselves provide the information when failures occur.

In Roos et al. [2002] a set S used to diagnose the multi-agent system is defined as S = (C, M, Id, Sd, Ctx, Obs), where C is a set of components, M is a specification of a possible fault in each component, Id is a set of identifiers of points that connect components, Sd is the description of the system, Ctx is a specification of input values of the system that are determined outside the system by the environment, and Obs is a set of observed values of the system. DRP-MAS follows a similar idea by extracting the necessary information to perform diagnoses from the information provided by the user.

Horling et al. [2000] examine the use of domain-independent diagnoses in multi-agent systems. Their approach is based on the assumption that the correct behavior, or at least the expected one, should be previously described. They have defined a goal/task decomposition language called TAEMS that can be used to describe goals and sub-goals. This language provides an explicit representation for goals, and pathways to achieve the sub-goals; each branch of the goal tree terminates at an executable method. It is possible to represent explicitly the expected behavior of the method, its expected quality, cost and duration, and also the other methods with which it interacts. In DRP-MAS the methods are represented by plans that are used to attain goals. Each plan has a set of possible related information, such as, the type and amount of resources used, desired goal, and expected quality. Therefore, if a plan does not achieve the agent's goal, it is possible to determine when and where the failure happened. Our approach offers a larger set of information than the one proposed by Horling et al. [2000], and also supports the choice of the agent's partners based on their reputations.

6 Conclusions

In this paper we have outlined the main challenges and related requirements as well as a design strategy to create a hybrid diagnostic-recommendation system for agent execution applied to a mobile process service. This system performs diagnoses and recommends alternative agent-based methods to achieve goals. The mobile process domain is used to illustrate our approach because it provides a representative set of scenarios.

Two important lessons were learned in the process of analyzing and developing the proposed system. At first, we realized that defining a universally efficient solution to perform diagnoses in different domains is extremely difficult, as different domains have characteristics which can significantly influence the result.

The second lesson is related to the use of the reputation concept. Knowing which agents are supplying information to the application agent that is asking for recommendations can be important because this information can be the reason for a failure in the execution of the plan. The *Recommendation agent* is able to select the agents with whom the application agent should interact by choosing the ones with the highest reputation.

We believe that the multi-agent systems approach is useful to resolve several current issues in ubiquitous computing. In future research we intend to determine the type of situations and problems that can occur when software agents are used in ubiquitous computing domains. We also intend to propose an ontology that can be used to describe these domains and their relationship to different devices.

References

BIGUS, J., BIGUS, J., Constructing Intelligent Agents Using Java, second edition, March, 2001.

BOELLA, G. AND TORRE, L., Regulative and Constitutive Norms in Normative Multi-Agent Systems. In Proceeding of 9th International Conference on the Principles of Knowledge Representation and Reasoning. California, 2004.

BRAUBACH, L.; LAMERSDORF, W., POKAHR, A., Jadex: Implementing a BDI Infrastructure for Jade Agents, Distributed Systems and Information Systems, University of Hamburg, September, 2003, vol 3, n. 3, 76-85, 2003.

CAIRE, G., 2003, LEAP User Guide, Copyright (C) TILAB, LEAP3.1, December, 2003.

DURAN, F., SILVA, V., *LUCENA*, C.J.P., Using Testimonies to Enforce the *Behavior* of Agents, In Proceedings of the *Coordination, Organizations, Institutions, and Norms in Agent Systems III.*, *COIN 2007, Springer-Verlag, 218-231*, 2007.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J., Design Patterns: Elements of Reusable Object-Oriented Software, 1994.

GUEDES, J., SILVA, V., *LUCENA*, C.J.P., A Reputation Model Based on testimonies, In Proceedings of the Agent Oriented Information Systems IV: Proc. of the 8th International Bi-Conference Workshop, AOIS 2006, Springer-*Verlag*, 37-52, 2006.

HORLING, B., LESSER, V., VINCENT, R., BAZZAN, A., XUAN, P., Diagnosis as an Integral Part of Multi-Agent Adaptability, In Proceedings of the DARPA Information Survivability Conference and Exposition, 2000, DISCEX'0, Volume 2, 211-219, 2000.

HUYNH, T. D., JENNINGS, N. AND SHADBOLT, N., *FIRE: an integrated trust and reputation model for open multi-agent systems*. In Proceedings of the 16th European Conference on Artificial Intelligence, 2004, Valencia, Spain, 2004.

HUYNH, T., JENNINGS, N. and SHADBOLT, N., Developing an integrated trust and reputation model for open multi-agent systems, In Proceedings of the 7th Workshop on Trust in Agent Societies, 2004.

JENNINGS, N., AND WOOLDRIDGE, M., Agent- Oriented Software Engineering; Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Multi-Agent System Engineering (MAAMAW-99), Vol. 1647, Springer-Verlag: Heidelberg, German,. 1-7, 1999.

JOHNSON, R., Building Application Frameworks: Object-Oriented Foundations of Framework Design (Hardcover) Wiley publisher, first edition, 1999.

KOOGAN, A. AND HOUAISS, A., Encyclopedia and Dictionary. Delta Publisher Rio de Janeiro, 1995.

LESSER, V., HORLING, B., and et al. The TAEMS whitepaper / envolving specification. http://dis.cs.umass.edu/research/taems/white/. Last access in November, 2007.

LI, T., PENG, Y., ZHAO, H., LI, K., Application of Multi-Agent in Control and Fault Diagnosis Systems. In Proceedings of the Third International Conference on Machine Learning and Cybernetics, August 2004, Shanghai, 26-29.

LOPEZ, F., Social Powers and Norms: Impact on Agent Behaviour. PhD thesis. University of Southampton. UK, 2003.

PATEL, J., TEACY, W., JENNINGS, N., LUCK, M., CHALMERS, S., OREN, N., NORMAN, T., PREECE, A., GRAY, P., SHERCLIFF, G., STOCKREISSER, P., SHAO, J., GRAY, W., FIDDIAN, N., THOMPSON, S., Monitoring, Policing and Trust for Grid-Based Virtual Organisations. In Proceedings of the UK e-Science All Hands Meeting 2005, UK, September 2005.

POUKAHR, A. AND BRAUBACH, L., Jadex User Guide, Distributed System Group University of Hamburg, Germany, Release 0.96, Available: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/. Last access in July, 2007.

POUKAHR, A. AND BRAUBACH, L., Jade Tool Guide, Distributed System Group University of Hamburg, Germany, Release 0.96. Available: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/. Last access in July, 2007.

POUKAHR, A. AND BRAUBACH, L., Jade Tutorial, Distributed System Group University of Hamburg, Germany, Release 0.96. Available: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/. Last access in July, 2007.

ROOS, N., TEIJE, A., BOS, A., CEES, W., An Analysis of Multi-Agent Diagnosis, In Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, AAMAS'02, 986-987, 2002.

SABATER, J. and SIERRA, C., REGRET: A Reputation Model for Gregarious Societies, In Proceedings of the 5th Int'l Conf. Autonomous Agents, ACM Press, 2001, 194-195.

SILVA, V., DURAN, F., GUEDES, J., LUCENA, C.J.P., Governing Multi-Agent Systems, In Journal of Brazilian Computer Society, special issue on Software Engineering for Multi-Agent Systems, n. 2 vol. 13, 19-34, 2007.

SILVA, V.; GARCIA, A.; BRANDAO, A.; CHAVEZ, C., LUCENA, C.J.P., ALENCAR, P., Taming Agents and Objects in Software Engineering, Software Engineering for Large-Scale Multi-Agent Systems, Springer-Verlag, 1999, 1-26.

SINGH, M., An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. Artificial Intelligence and Law v. 7 (1), 97-113., 1999

SMITH, J., The publishing company. London, 2nd edition, 1998.

University of Hamburg, Distributed Systems and Information Systems Group, Germany, Acceded at: *http://vsi-www.informatik.uni-hamburg.de*, 2008.

VINCENT, R.,, HORLING, B., Experiences in Simulating Multi-Agent Systems Using TAEMS, 2000, The Proceedings of the 4th International Conference on MultiAgent Systems, 2000,455-456.

WAGNER, T., VALERIE, G., PHELPS, J., TAEMS Agents: Enabling Dynamic Distributed Supply Chain Management, ThElectronic Commerce Research and Applications, Volume 2, Number 2, 114-132, 2003.

WOOLDRIDGE, M., AND CIANCARINI, P., *Agent-Oriented Software Engineering: The State of the Art, in* First Int. Workshop on Agent-Oriented Software Engineering, Vol. 1957, Springer-Verlag, Berlin, 1-28, 2000.