A Hybrid Diagnostic-Recommendation System for Agent Execution Applied to Ubiquitous Computing Systems

Andrew D. Costa¹, Carlos J. P. Lucena¹, Viviane T. Silva²,
Donald Cowan³, Paulo Alencar³, Baldoino Fonseca¹

¹Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro - RJ – Brazil

²Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid (UCM)
Madrid, Spain.

³School of Computer Science
University of Waterloo
Waterloo, Canada.

{acosta, lucena, bneto}@inf.puc-rio.br,
{dcowan, palencar}@uwaterloo.ca

viviane@fdi.ucm.es

Abstract When agents of software based systems operate in some domains such as ubiquitous computing, they may not be able to attain their goals owing to failures during system execution. 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. Sometimes, the problem relates to the information exchanged among the software agents, which may depend on their levels of trust and reputation. In this paper, we discuss solutions to the main challenges of creating diagnoses and provide recommendations about agent execution to support eventual attainment of goals in ubiquitous computing systems. We also propose a hybrid diagnostic-recommendation framework that provides support for different methods of addressing such challenges and we demonstrate how the framework can be used in ubiquitous computing applications.

1. Introduction

Multi-Agent Systems (MAS) [Jennings and Wooldridge 1999] organized as collections of autonomous and heterogeneous agents working together to achieve a set of goals is a new paradigm for the software engineering of complex and distributed systems. However, in many cases the agents are not able to attain their goals owing to failures during system execution. When an agent tries to achieve its desired goal, but faces this type of failure, it becomes relevant to understand why such failures occurred and what can be done to remedy the problem.

Interesting applications, which illustrate this idea, are often 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. If we consider these service requests to be provided by agents from different devices, i.e., by heterogeneous agents potentially designed and developed by different developers, failures may occur. Thus, to diagnose the problem and provide associated recommendations to agents to attain their original goals is fundamental to the agents’ execution.

Some proposals that have appeared in the literature suggest different ways for agents to diagnose system execution failures. In the paper [Horling et al. 2000], the authors examine how a domain independent diagnoses can behave in multi-agent systems. This paper provides some very useful concepts related to diagnosis, which include the provision of techniques to determine the correct behaviour of an agent, and to compare this result with the actual result. However, the approach does not offer a significant set of data to determine the expected behaviours, and whether the reason for some failure was some agent, which provided a bad information. Related with this idea, we have the reputation concept, which we consider as a social notion associated with the observed trustfulness of its individuals [Koogan et al. 1995]. Therefore, knowing an agent’s reputation can be very useful in choosing an agent for the next transaction, because some information provided from someone can be the reason to a goal not to be achieved.

In this paper we describe a new hybrid diagnostic-recommendation system for agent execution applied to ubiquitous applications that does not violate an agent’s privacy. Diagnosis is defined as the process of determining why agents did not achieve their goals. Recommendations are then provided on how to achieve these goals from alternative strategies, such as, other plans of execution interacting with alternative agents, etc.

Our hybrid diagnostic-recommendation framework called DRP-MAS (Diagnosing and Recommending Plans in Multi-Agent Systems) proposed in this paper can be instantiated to perform different kinds of diagnoses and to provide recommendations in assisting an agent to achieve its desired goals.

This paper is structured as follows. In Section 2 we discuss some of the main difficulties of diagnosing 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 simple case study involving ubiquitous computing and Section 5 contains conclusions and possible future directions for the research.

2. Diagnosing and Providing Recommendations in Ubiquitous Computing Domains

In this section, we analyze the main requirements that were used for the design of the DRP-MAS framework in the context of ubiquitous computing. 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 as requirements are presented in the next few paragraphs.
Deciding how to analyze the behaviour of agents: The first challenge focuses on the analysis of agent behaviour. Two approaches were considered. In the first, the execution of each agent was monitored [Li et al. 2004]. However, as we are working in a multi-agent system environment using heterogeneous agents, it appears their privacy would be violated. In the second approach, each agent analyzes its own behaviour during execution. In this case, the agent gathers useful information to be used in the future.

Selecting data for diagnosis: A second challenge relates to choosing the appropriate data to perform diagnoses on the execution behaviour of an agent. To execute a diagnosis, different information might be used, such as, the record of success or failure of communications with another agent, and the description of the problem arising from availability of resources, for instance, insufficient memory space. A list with such data is identified and recorded.

1. Determining strategies for diagnoses: As different domains can provide specific information to make diagnoses, the strategy used in each domain can be different. Thus, the challenge is to find a coherent approach for defining services or strategies to be used in different domains and to define a flexible approach that can be adapted to different domains.

Determining trustworthy agents: When a plan is executed by an agent, it may be necessary to negotiate with other agents. When that negotiation occurs, the information received by an agent can determine whether it will be successful on achieving its goal. Therefore, when a diagnosis is formulated and it is verified that a specific agent was involved in the unsuccessful execution then it may be appropriate to associate a bad reputation to (or to decrease the reputation of) such agent in order to prevent future interaction with it. Reputation is a social notion associated with the observed trustfulness of its individuals. However, providing criteria to distinguish the degree of trust an agent may have on another is often a difficult task. Trust is seen in this case as the amount of faith one is willing to assign to someone else’s integrity.

Determining strategies for recommendations: As well as the strategies used for providing diagnoses, the strategies for generating the recommendation may be domain dependent since domain dependent data can determine the recommendation to match the diagnosis. Therefore, the approach we propose can be instantiated to different domains.

Different devices: Limitations of hardware 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.

Representing profiles of agents: Agents can have profiles which define important characteristics, such as the agent’s reputation relative to negotiations. The challenge
is how to represent profiles of agents, and how they can influence the recommendations provided by the application instantiated by the framework.

3. The DRP-MAS framework

In this section, we present the DRP-MAS framework (Diagnosing and Recommending Plans in Multi-Agent Systems). The framework helps to perform different diagnoses and provide recommendations for agents in achieving their goals. First, an outline of the framework is presented, followed by its architecture, and then some important concepts on which it is based.

3.1 The General Idea

The DRP-MAS framework is used when an agent involved in an application does not achieve one of its goals after the execution of one of its plans. The application agent that is not successful, called the Requester agent, sends a request to the Mediator agent that creates a specific Diagnostic agent (responsible for providing diagnoses) and a Recommendation agent (responsible for providing recommendations). After that it sends a message to the Requester indicating which Diagnostic agent to use (Figure 1).

Once the specific Diagnostic agent is created and identified, the Requester requests advice from the Diagnostic agent in order to achieve a desired goal (Figure 1). For that purpose, it sends a message to the Diagnostic agent with the values of a set of attributes that can help in the analysis, such as: plan executed, goal not achieved, the agents used in the negotiations, its profile, and a number that represents the quality of the execution performed (details shown in Section 3.3). This idea of a quality number was based on the work reported in [Horling et. al 2000] and [Lesser et al 2007].

When the Diagnostic agent receives the message, it tries to find the reason that the Requester agent did not achieve the desired goal. At the end of the analysis, the Requester 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 detect the reason why the Requester agent did not achieve the desired goal. In this case,
when the Recommendation agent receives the message indicating that it was not possible to provide a diagnosis, it still tries to select plans that can be used to achieve the desired goal.

In the case in which a diagnosis is provided, the Recommendation searches through alternative plans to achieve the goal (details shown in Section 3.4) using the data in the diagnosis. 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. This analysis is based on the services used in the plan.

Using the set of agents that can perform the same services in support of the interactions, the Recommendation agent then uses each of these agent’s reputations to select the “best” agents. The profile of the Requester agent can be an important piece of information to define which agents should be selected. When the Recommendation agent finishes, a message is sent to the Requester agent with the recommendations.

Before a Requester requests recommendations from the agents of the DRP-MAS, the relation of the plans that can be recommended together with the services used, must be provided to the framework. To achieve this result, some agent must supply the previous relation between plans and other data. Therefore when some diagnosis or recommendation is performed, the plans should have been configured. Such plans of recommendation are available in a Plan base which Recommendation agents with different strategies can access.

### 3.2 Architecture

In this section we describe the architecture of the system used in our approach. As illustrated in Figure 2 the DRP-MAS sub-system communicates with both the Application sub-system and the Reputation sub-system. The DRP-MAS is composed of four modules: Mediation, Diagnosis, Recommendation and Artificial Intelligence Toolset.

![Figure 2 Architecture](image)

The mediation module is responsible for providing the Mediator agent (Section 3.1), which creates a Diagnostic agent and a Recommendation agent corresponding to a Requester agent defined in the Application. 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 defines an API
(Application Programming Interface) called Bigus [Bigus 2001], which supports different types of reasoning algorithms (forward chaining, backward chaining and fuzzy logic) to perform the various processes.

The Reputation sub-system supplies reputations to the DRP-MAS and to the Application sub-system when requested. In the current implementation, we created this model based on the Fire model [Huynh et al 2004] and based on the Global reputation concept presented by the Governance Framework [Silva et al. 2007]. To follow the main modules of the DRP-MAS are better explained, Diagnosis and Recommendation, besides the Reputation sub-system.

3.3 Diagnosis Module

The diagnosis described in section 3.1 is performed by the Diagnostic agent provided by the framework. Such diagnoses are based on data provided by the Requester agent and different data can be required by different domains. Nevertheless, it is possible to state a set of domain-independent data that can be used by such strategies while supplying the diagnoses. Note the data used to provide diagnoses can also be used to provide recommendations, as explained in Section 3.4. In what follows, the data provide for diagnosis is detailed below.

1. Resources and associated problems - In [Horling 2000] resources are considered important data to support diagnoses. A possible reason for an execution failure can be the absence or an insufficient amount of a resource to perform an activity.

2. Quality of execution - Assigning a quality factor to an activity performed by an agent as defined by the TAEMS model [Lesser et al 2007] can be useful in diagnosis. This factor can represent different information, such as indicating whether all the steps executed by an agent were successful.

3. Goal - The execution of an agent’s plan is always associated with a goal that the agent wants to achieve. Knowing this goal is fundamental to provide both a diagnosis and corresponding advice. The advice about other plans to be executed will be based on this 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 attempted by the Requester agent.

5. Agents with whom the agent interacted - The diagnosis may indicate that some agent has not provided a service. Thus, it is important to know which agents were involved during the execution of the plan by the Requester agent.

6. Services used – The services used during an execution can be useful in providing advice about agents that might provide alternative services.

7. Profile – Each agent has a profile, which can represent some of its characteristics. In a profile, we can stipulate the minimum acceptable degree of reputation for the agents that provide information to the Requester agent. This information can be
useful in the process of providing recommendations performed by the Recommendation agent, where the choice should follow the definition represented in the profile of the Requester.

8. Device used – Depending on the device used to request a service, a different format of data must be provided during execution of a plan. The data provided to the plan about the device are: (i) type of device (ex: cell phone, laptop, etc), (ii) model of the device (ex: LG MG296 GSM, etc.) and (iii) the language in which the data must be provided by the agent (ex: English, Portuguese, etc).

9. Connections – Data about the connection used by a device can be crucial to determine a recommendation or to meet a diagnosis. To identify the different types of connection the following information was defined: (i) speed of connection (ex: 56Kbps, 2Mbps, etc), (ii) technology of communication (ex: wireless, LAN, WAN, etc.) and (iii) IP address.

10. Request information– Each request performed by some agent has important identifying data: (i) request identifier, (ii) date and hour of the request and (iii) grade of severity of the request.

11. Belief Base – The knowledge base used by an agent including the time of its last update can be useful to perform diagnoses and to provide recommendations.

12. Problems met by the Requester - The quality of service offers a list of elements representing other problems that were not mapped into the previous information. These problems can also be used for 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 [Fayad et al 1999] (flexible points) to support flexible definition of diagnosis strategies. Note that different strategies can be used by the same Requester agent depending on the data available after the 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 toolset module defined in the framework. The strategies use the BIGUS API to access such algorithms.

The DRP-MAS focuses on the forward chaining algorithm. We believe that this approach is very useful for performing diagnoses, as this algorithm uses inference rules from a set of available data in order to extract more data while seeking an optimal goal. This approach seems appropriate since a Requester agent provides a set of data to the Diagnostic agent. With this data and forward chaining it is possible to define a rule base, which allows a diagnosis to be inferred from the data provided by a Requester.

Different diagnoses can be defined using the same rule base. For instance, a previously successful diagnosis can be used, especially when little data are provided by the Requester. Imagine that Diagnosis A was provided from a set of data. Our framework verifies which other diagnoses could be used if more data had been provided by the Requester agent. First, the framework analyzes the data used to meet Diagnosis A, and using the forward chaining rule it verifies which data were not provided that
could be useful to meet new diagnoses. Then, the DRP-MAS provides values for the data that were not used, in order to provide diagnoses that better define the reason that an agent did not achieve a desired goal. These values are defined from the conditions of the rule base, which must be defined by the application. Figure 3 illustrates a possible representation of relations between diagnoses. In this case, Diagnosis A was reached from the data provided by the Requester, while four new diagnoses were inferred from data that were not provided: Diagnoses B, C, D and E. The leaf nodes, C, D and E, represent diagnoses more detailed than the Diagnosis A and the diagnoses provided to the Recommendation agent are the leaf nodes describing in more detail the possible reasons for a goal not to have been attained. We believe that this approach is mainly useful when some Requester does not decide to provide some data that it considers useless, or when it forgets of providing it.

![Figure 3 Relation of diagnoses](image)

### 3.4 Recommendation Module

The **Recommendation agent** includes the process of providing alternative ways (recommendations) to achieving a goal, which is composed of three steps: selecting plans, verifying the plan’s needs for agents to request information, and choosing appropriate agents.

The first step is executed when the **Recommendation agent** receives the diagnosis from the **Diagnostic agent** and verifies which plans can be used to achieve a specific goal. From the diagnoses and data provided by the **Requester agent**, a strategy defined by the application can use a service offered by the DRP-MAS that supports searching plans related to such data. In order to represent this idea, two search templates were defined: acceptable data template and refused data template. Each one has the data of the information set described in sub-section 3.3. The first template defines the data that must be related to a selected plan. The second template defines the data that must not be related to a selected plan. Therefore strategies can specify plans that are related to data X and not data Y. In the end of this step the **Recommendation agent** verifies the plans to be chosen: if no plan is chosen, then a message is sent to the **Requester**, otherwise, the second step is executed.

The second step verifies if a selected plan needs the assistance of agents in order to request information. If it is not necessary, then the process is finished and a message containing the recommendations is sent, otherwise, the **Recommendation agent** requests reputations of the candidate agents selected to some agents of the application. In the final step, the **Recommendation agent** receives the reputations in order to recommend the agents to the plans chosen.
3.5 Reputation Sub-System

The Reputation sub-system offers two reputation models: centralized and decentralized. The first is represented from the Global reputation concept used by the Governance Framework [Silva et al. 2007]. The reputations are shared among the agents of the application and managed for a Reputation agent, which is responsible for receiving the requests to provide the desired reputations, besides receiving the new reputations calculated from agents in order to update the global reputations of the application.

The second model, decentralized, was implemented by the use of Fire [Huynh 2004]. From the set of available trust and reputation types proposed by the reputation model, our approach uses the following:

- interaction trust (resulting from past experiences from direct interactions),
- witness reputation (reports of witnesses about an agent’s behaviour),
- and certified reputation (references provided by other agents about an agent’s behaviour).

We have chosen Fire since it provides not only the interaction trust and witness reputations, also available in other reputation models, but also because it provides certified reputations. Such reputations are fundamental when an agent wants to know the reputation of other agents with whom it has not interact with and when it does not know any other agent that has interact with the desired one. In next section we present case study, which uses the global reputation, in order to illustrate an example how the DRP-MAS framework can be used.

4. Case Study: Translation

In this 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 Figure 4: (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 4). 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.

![Figure 4 Relation between the diagnoses and their respective plans](image)

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 5 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 5 Translator agent updating belief base](image)

5. Conclusions and Future Works

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


