Massimo Cossentino
Michael Kaisers
Karl Tuyls
Gerhard Weiss (Eds.)

LNAI 7541

Multi-Agent Systems

9th European Workshop, EUMAS 2011
Maastricht, The Netherlands, November 2011
Revised Selected Papers

Lecture Notes in Artificial Intelligence 7541

Subseries of Lecture Notes in Computer Science

Massimo Cossentino Michael Kaisers
Karl Tuyls Gerhard Weiss (Eds.)

Multi-Agent Systems

9th European Workshop, EUMAS 2011
Maastricht, The Netherlands, November 14-15, 2011
Revised Selected Papers

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Massimo Cossentino
ICAR-CNR, Palermo, Italy
E-mail: cossentino@pa.icar.cnr.it

Michael Kaisers
Maastricht University, Department of Knowledge Engineering
Maastricht, The Netherlands
E-mail: michaelkaisers@googlemail.com

Karl Tuyls
Maastricht University, Department of Knowledge Engineering
Maastricht, The Netherlands
E-mail: k.tuyls@maastrichtuniversity.nl

Gerhard Weiss
Maastricht University, Department of Knowledge Engineering
Maastricht, The Netherlands
E-mail: gerhard.weiss@maastrichtuniversity.nl

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-34798-6 e-ISBN 978-3-642-34799-3
DOI 10.1007/978-3-642-34799-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012951794

CR Subject Classification (1998): I.2.11, I.2, H.3, H.4, C.2.4, D.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

In the last two decades, we have seen a significant increase of interest in agent-based computing. This field is now set to become one of the key technologies in the twenty-first century. It is crucial that both academics and industrialists within Europe have access to a forum at which current research and application issues are presented and discussed.

In December 2003, the First European Workshop on Multi-Agent Systems (EUMAS) was held at the University of Oxford, UK. This workshop emerged from a number of related workshops and other scholarly activities that were taking place at both national and European levels, and was intended to provide a single recognized forum at which researchers and those interested in activities relating to research in the area of autonomous agents and multi-agent systems could meet, present research results, problems, and issues in an open and informal but academic environment. This set-up allows for discussions of the latest, potentially preliminary findings of state-of-the-art research.

Following in the tradition of past EUMAS events (Oxford 2003, Barcelona 2004, Brussels 2005, Lisbon 2006, Hammamet 2007, Bath 2008, Agia Napa 2009, and Paris 2010), the aim of the 9th European Workshop on Multi-Agent Systems held in Maastricht (The Netherlands) during November 14–15, 2011, was to encourage and support activity in the research and development of multi-agent systems, in academic and industrial efforts.

In 2011, the EUMAS workshop had 45 papers accepted for oral presentation and the demo session hosted four contributions. The workshop was very well attended and the presentations gave the opportunity for several debates often continuing during the coffee breaks.

The two-day event also gave the opportunity to listen to three exciting invited talks given by Marie-Pierre Gleizes (Université Paul Sabatier, France), Peter McBurney (King's College London, UK), and Milind Tambe (University of Southern California, USA).

After the workshop, the best papers were selected taking into account the reviews they received during the pre-workshop review phase and the discussion generated by their presentations. The authors of these papers were asked to significantly improve their manuscripts and the resulting works passed through a new review cycle. These papers are the backbone of this book. They are perfectly completed by three papers written by the invited speakers, discussing key issues in multi-agent systems.

May 2012

Massimo Cossentino
Michael Kaisers
Karl Tuyls
Gerhard Weiss

Improving Diagnosis Agents with Hybrid Hypotheses Confirmation Reasoning Techniques

Álvaro Carrera and Carlos A. Iglesias

Universidad Politécnica de Madrid,
Madrid, Spain
{a.carrera,cif}@dit.upm.es

Abstract. This article proposes a Multi-Agent Systems (MAS) architecture for network diagnosis under uncertainty. Network diagnosis is divided into two inference processes: hypotheses generation and hypotheses confirmation. The first process is distributed among several agents based on a Multiply Sectioned Bayesian Network (MSBN), while the second one is carried out by agents using semantic reasoning. A diagnosis ontology has been defined in order to combine both reasoning processes. To drive the deliberation process, the strength of influence obtained from Cumulative Distribution Function (CDF) method is used during diagnosis process. In order to achieve quick and reliable diagnoses, this influence is used to choose the best action to perform. This approach has been evaluated in a P2P video streaming scenario. Computational and time improvements are highlighted as conclusions.

Keywords: agent, Bayesian, ontology, diagnosis, network.

1 Introduction

The complexity of telecommunication networks has increased the demand for network and service management systems. Nowadays, network fault management requires high skilled engineers, which are not able to cope with the increasing heterogeneity and complexity of the network. The probability of occurrence of faults in large telecommunication networks grows as they become widespread, complex and heterogeneous [3]. Thus, the role of automatic diagnosis modules is getting more attention, in order to cover faults detection, isolation and recovery.

Furthermore, other important aspect to point out is the need for dealing with uncertainty during the diagnosis task, since many corroboration tasks cannot be carried out because of different reasons, such as the cost itself of the action or that the action requires to access the subscriber equipment and could cause him any trouble.

In recent past, several works have studied different approaches to deal with uncertainty using Bayesian networks for diagnosis [7,11]. The main focus of this work is to present a MAS architecture that combines two reasoning processes: semantic reasoning and Bayesian reasoning. This approach proposes to use Bayesian inference to handle uncertainty inherent in any diagnosis process and

M. Cossentino et al. (Eds.): EUMAS 2011, LNAI 7541, pp. 48–62, 2012.

semantic inference to discriminate which action is the best one to perform depending on the available data.

The reminder of this article is structured as follows. Firstly, Sect. 2 shows the knowledge model used in this work. Sect. 3 proposes an agent architecture for reasoning during both phases of a diagnosis: hypotheses generation and hypotheses confirmation. Sect. 4 exposes the testbed scenario and exemplifies the diagnosis process. Sect. 5 shows the evaluation and presents the results of comparison with other approaches. Finally, Sect. 6 draws out the main conclusions about the application of this approach and, besides, a brief description of future possible improvements.

2 Knowledge-Level Model of the Diagnosis Task for Telecommunication Networks

Following the knowledge-level analysis of the diagnosis task by Benjamins [2], diagnosis can be decomposed into three subtasks: (i) *symptom detection*, finding out whether complaints are indeed symptoms, (ii) *hypotheses generation*, generating possible causes based on the symptoms, and (iii) *hypotheses discrimination*, discriminating between the hypotheses based on additional observations.

In this article, we focus on the last two tasks, hypotheses generation and discrimination, as well as in the repair task, as illustrated in Fig. 1.

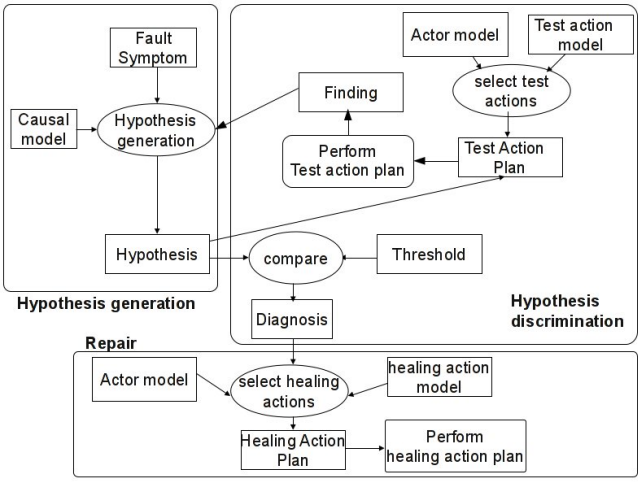

Fig. 1. Diagnosis inference structure. Legend: box (concept), oval (inference), rounded corner box (task).

The first process, *hypotheses generation*, consists of generating hypotheses from the notified fault based on a causal model. Since this process needs to handle uncertainty, a Bayesian network has been selected for expressing the

causal model. Moreover, given that this Bayesian network could not scale well with the size and heterogeneity of telecommunication networks, our architecture proposes the usage of MSBN [13] technique, which allows to distribute this reasoning process across MASs.

The two other processes, *hypotheses discrimination* and *repairing*, follow a similar pattern. The first one obtains a test action plan to confirm the generated hypotheses. This process contains a list of ordered actions to be executed based on the expected benefits of the tests. The expected benefit of an action is defined as how relevant is for the current and is equal to the influence between variables inside causal model [8]. In this way, the system can perform more efficient hypotheses discrimination (as shown in Sect. 5).

Finally, the repair process obtains a healing action plan to repair the confirmed diagnosis. In order to reason under uncertainty, we propose to use an ontology based reasoning process, combining a diagnosis ontology expressed in Ontology Web Language (OWL) [14] with rules expressed in Semantic Web Rule Language (SWRL) [10].

Nevertheless, a technique to communicate both reasoning processes is needed, in order to be able to provide feedback and integrate learning mechanisms of the confidence of the generated hypotheses based on the results of the tests.

In order to carry out the exposed diagnosis process, an upper-ontology has been defined to facilitate the communication between the agents in the fault diagnosis task. This upper-ontology (Fig. 2) shows that *hypotheses* are generated according to *failure classes*. These hypotheses identify a suspected *component* as the *location* of the failure. In this way, the ontology represents what is happening and where is happening. Depending on the hypothesis class, different *actions* can be carried out for corroborating the hypothesis (*test actions*) or repairing the component (*healing actions*). All *actions* have *conditions*(preconditions and postconditions) that allow somebody to evaluate its eligibility for execution.

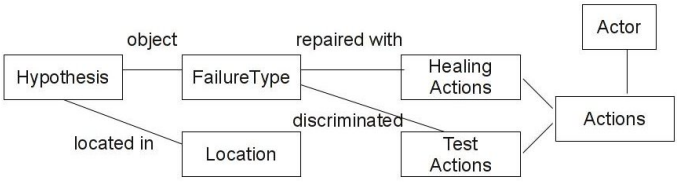

Fig. 2. Upper-ontology for diagnosis

Furthermore, the upper-ontology also includes the concept of *diagnosis*. A *diagnosis* has its set of hypotheses, its set of performed tests and its set of performed healing actions. This collection of data is useful for self-learning processes as reinforcement learning, for example which healing actions repaired a certain failure.

Another important assumption is that *actions* are executed by *actors*. *Actors* can be *humans* (manual actions) or *agents* (automatic actions). *Actions* can be

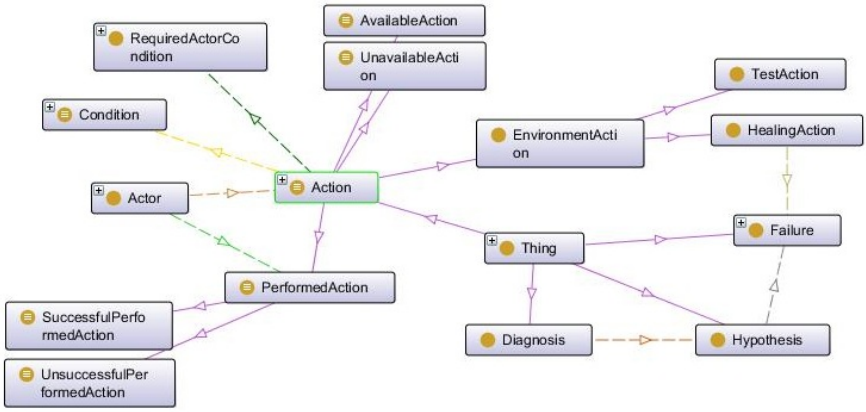

Fig. 3. Action upper-ontology

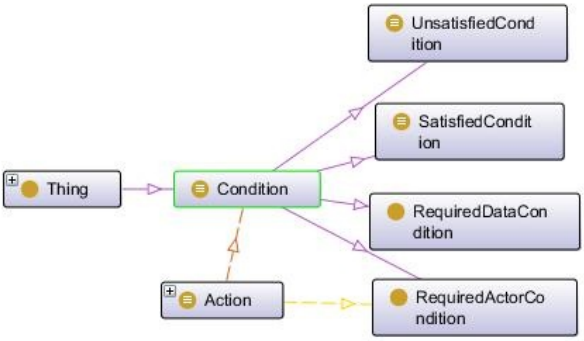

Fig. 4. Condition upper-ontology

classified according to the disjoint classes *Available / Unavailable*, when all preconditions are satisfied or not; and, if an *available action* has been performed, it is classified according to the disjoint classes *Successful Performed / Unsuccessful Performed*, when all postconditions are satisfied or not. This classification is shown in Fig. 3.

This model has been formalised as an OWL ontology for reasoning on diagnosis tasks. To adapt this generic diagnosis ontology for a specific diagnosis case, the generic diagnosis ontology must be extended with specific concepts, i.e. possible faults, specific actors, specific conditions, etc. The conditions of *an action* are modelled with the ontology class *Condition*. Fig. 4 shows two generic conditions: *Required Data* that specifies a required parameter and *Required Actor* that specifies an actor to perform the action. The second one is condition for all actions, because all actions need to be executed by someone. But these two conditions are generic conditions; all conditions that particularise specific restrictions should be

added depending on the domain. *Conditions* can be classified according to the disjoint classes *Satisfied/Unsatisfied*. To check all conditions, we use SWRL. In section 4, the use of *conditions* and SWRL rules is shown.

3 Agent Architecture for Diagnosis

The upper-ontology presented in Sect. 2 has guided the design of an agent architecture that performs out both tasks: hypotheses generation and hypotheses confirmation.

The proposed agent architecture (Fig. 5) consists of four modules. Hypotheses generation is carried out by *Bayesian module* and the Hypotheses confirmation, by *Ontology module*. Both modules are governed by *Agent Control module* which is an extended Belief-Desire-Intention (BDI) agent architecture (B2DI Agent Model [4]) where beliefs are distributed and shared across the MAS. *Bayesian Module* is a reasoning inference engine that processes environment data to infer possible fault root causes (i.e. hypotheses with associated confidences). For this task, Bayesian networks are used to represent several concepts like symptom, possible root causes, etc. and the relations among them in a Directed Acyclic Graph (DAG) in which each node contains a Conditional Probability Table (CPT). Bayesian networks have been design accordign to BN3M model [9] which structures causal models in Bayesian networks to three groups of notes: *context*, *fault* and *evidence*. *Context* variables model the environment, in this case, these variables are used to model information about the network in which each agent resides. *Fault* and *Evidence* variables are used to model the possible failures through hypotheses and observations (i.e. results of tests).

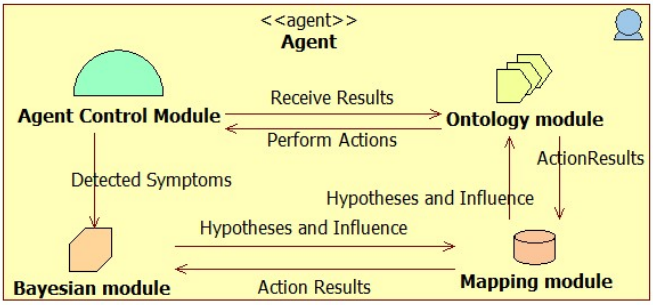

Fig. 5. Agent Architecture

Since a Bayesian network is a DAG with probability distributions, each node has a concrete influence on its neighbours [8]. The relevance of this influence in the diagnosis procedure varies depending on the available information about the environment. In other words, the influence represents how useful is the information that the agent would obtain if it would perform one action, for example,

if it would execute a test. To obtain these data, CDF [9] distance is used. The outcomes of this method are used to sort all possible actions for an agent by relevance in order to reach a reliable and fast diagnosis.

The outcomes of hypotheses generation task is added to *Ontology-based Reasoning Module*. This module is responsible for deliberating which action should be performed out using the outcomes of the *Bayesian module*. This module filters the sorted action list based on preconditions of each action (see Sect. 2). After executing one action, its result is fedback to the *Bayesian module* to generate updated hypotheses.

It should pointed out that the mapping between *Bayesian module* and *Ontology module* is not trivial. So, *Mapping module* performs the mapping process to create ontology individuals and extract information from ontology concepts to probabilistic data that can be input to the Bayesian module. To perform this task, we use Probabilistic OWL (PR-OWL) [5] ontology that supports a way to add probabilistic information to others concepts defined using OWL.

SWRL rules are used before, during and after diagnosis process to choose which action is the best one to perform in each moment to diagnose a problem or to fix it, to check if the result of a performed action is the expected one or it was an error, to notify human operators, etc. Finally, the use of SWRL, OWL and Bayesian networks adds adaptability to the system since the behaviour of the agent can be deployed by a simple message with an OWL file as content that adds or modifies the current rules or Bayesian networks on the fly using PR-OWL.

Since presented modules can be split in different agents, some functionalities can be distributed across several agents in order to obtain more scalability, remote access to restricted data, less computational requirements, etc. Depending on which modules compound each agent, we can classify agents in three types:

- *Fully Autonomous Agent* which has all modules presented before. It is able to evaluate the environment, reason (in a distributed way) under uncertainty, perform actions, etc. It can work autonomously, but it has better performance working together with other agents.
- *Semi-Autonomous Agent* which has Agent Control Module and Ontology based Reasoning Module. It cannot deal with uncertainty, but it is able to interact with its environment. To reason with uncertainty, it has to interact with an *Fully Autonomous Agent*.
- *Dependent Agent* which has only the Agent Control Module. It is able only to perform prefixed request actions. For example, the execution of one test or one monitoring action.

Furthermore, the extensibility of the upper-ontology is possible using specific domain ontologies and Bayesian networks which represent diagnosis knowledge of a diagnosis domain.

Uncertainty handling and extensibility are highly recommended features for systems that work in complex environments like network management. Our proposal consists of defining a flexible agent architecture which integrates the previously identified modules. These functionalities can be distributed at design time

or even run time by the agents themselves (creating agents on demand), depending on non functional requirements (time restrictions) or functional requirements (distribution requirements for complex actions on remote equipments).

4 Case Study

This section shows the case study used in this work. First of all, the scenario used to evaluate the model is presented in Sect. 4.1. Sect. 4.2 presents a example of the proposed agent architecture. In Sect. 4.3, the agents deployment is exposed in order to facilitate the explanation of a detailed diagnosis case, shown in Sect.4.4.

4.1 Scenario

To properly frame this study, a P2P streaming scenario (see Fig. 7) was chosen. In this scenario, there are a multimedia provider user and a multimedia consumer user. Multimedia contents are stored in a video server inside of the Multimedia Provider Home Area Network (HAN) and are remotely accessed from the Multimedia Consumer HAN. Multimedia contents are transmitted in real time using Real Time Streaming Protocol (RTSP) for session establishment and Real-time Transport Protocol (RTP) for content delivery. Many faults may occur both in connection and in services. The system is designed to provide, to an end-user or an operator, the result of the diagnosis made upon receipt of a failure symptom notification. The diagnosis result is expressed in percentages representing the certainty of the occurrence of a given hypothesis.

4.2 Agent Architecture Example

For exemplification purposes and facilitating the understanding of the deployment of agents in the scenario, an agent responsible to diagnose faults in Multimedia Provider HAN is presented.

Agent Control Module has a main goal that is to diagnose network faults. This module is responsible for acting as bridge between the other modules. *Bayesian module* contains a Bayesian network that models possible failures and possible tests in the Provider HAN region. A simplified version of this Bayesian network and one of its CPTs are shown in Fig. 6.

Ontology based Reasoning module works with a specific domain diagnosis ontology specialised for P2P streaming scenario. In other words, this ontology that extends the generic diagnosis ontology presented previously contains specific concepts like *Session* or *RTPMonitoringAction*, specific conditions like *RequiredRTPSessionCondition*, etc. *Mapping module* has been adapted to properly translate data between both domains: semantic and probabilistic domains.

4.3 Agents Deployment

In the scenario presented in Sect. 4.1, some agents have been deployed according to their geographic distribution for exemplification purposes. One *Fully Autonomous Agent* is executed inside each subnetwork. One agent has been deployed

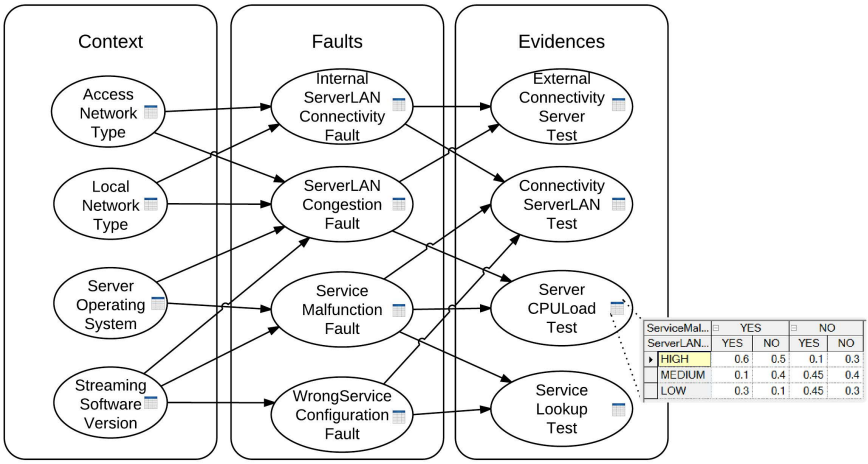

Fig. 6. Bayesian network for case study

in a internal server of ISP Network. The other two ones have been deployed in HAN gateways. Each one of these three agents is responsible to diagnose faults inside their domain. These agents have been deployed into devices with enough computational resources, but other three *Semi-Autonomous Agents* has been deployed taking care of minimising the consumed resources, since they are deployed in final user domain, such as multimedia consumer PC. One of them resides into the multimedia consumer PC. This agent has monitoring capabilities to detect and monitor quality of streaming sessions. The second one, into Multimedia Streaming Server with low resources. This agent has testing capabilities to know the server status. The other one is deployed in the network operator terminal to notify diagnosis results. These deployments are shown in Fig. 7.

Notice that these agents publish all actions they are able to perform in a directory facilitator like a service. Thus, any agent can request an action execution.

4.4 Streaming Diagnosis Case

In order to facilitate the understanding of the example, only three agents are involved in the exposed diagnosis case: "StreamingClientAgent" (Semi-Autonomous Agent), "DiagnosisClientAgent" and "DiagnosisServerAgent" (both Fully Autonomous Agents). Notice that all words with *italic* style in this section represent ontology classes, not ontology individuals.

The presented case study starts when a user requests a video streaming session. This streaming session is detected by the Semi-Autonomous Agent that resides inside multimedia client PC, named "StreamingClientAgent". This agent creates a new *RTPSession* individual in the ontology with properties that represents information about this session, like, which computer is the client and which computer is the streaming server. The creation of this individual triggers

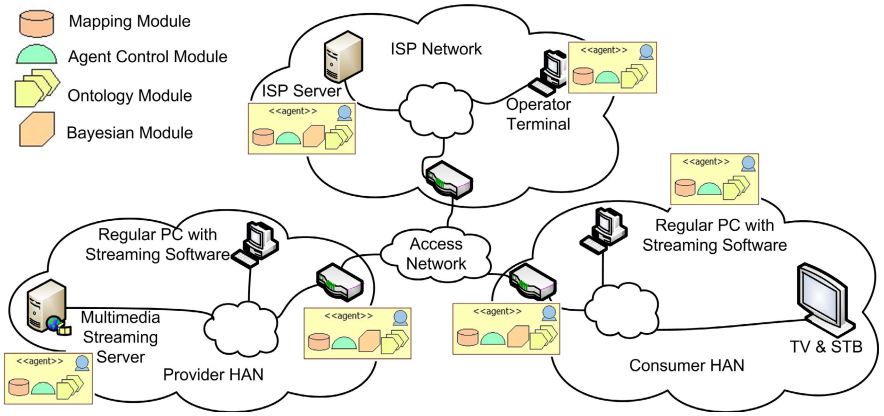

Fig. 7. Agent deployment in case study scenario

SWRL engine to evaluate if any possible action is now available, i.e. if all its preconditions are satisfied.

To simplify the explanation, we consider this agent is able to do only two types of actions: *RTPMonitoringAction* (to monitor Quality of Service (QoS) in RTP sessions) and *NotifySymptomAction* (to notify other agents about a new detected symptoms).

RTPMonitoringAction has two preconditions: *RequiredActorCondition* and *RequiredRTPSessionCondition*. To check if this conditions are satisfied, several SWRL rules are used. Rule 1 is used for *RequiredActorCondition* and Rule 2 for *RequiredRTPSessionCondition*.

Rule 1. Actor(?actor),
RequiredActorCondition(?condition),
hasPrecondition(?action, ?condition),
canPerform(?actor, ?action)
→ satisfied(?condition, true)

Rule 1 searches all individuals of class *Actor* and class *RequiredActorCondition*. Then, it searches all actions which have one condition of this type. And finally, it searches if one agent can perform a determined action. If these subconditions are satisfied, the *RequiredActorCondition* individual changes its property "satisfied" to true.

Rule 2. RTPSession(?session),
RequiredRTPSessionCondition(?condition),
hasPrecondition(?action, ?condition),
hasClient(?session, ?system),
Actor(?agent),
id(?agent, ClientAgent),
isExecutingIn(?agent, ?system),
→ satisfied(?condition, true)

Rule 2 searches all individuals of class *RTPSession* and class *RequiredRTPSessionCondition*. Then, it searches all actions which have one condition of this type. It obtains the client system of the streaming session and gets the local system, i.e. the system in which the agent is executing currently. If both systems are the same, the condition is satisfied.

Once both conditions are satisfied, the *Action* individual changes its property "available" to true. Since "StreamingClientAgent" does not have more available actions, *RTPMonitoringAction* is selected to be executed. So, it performs a monitoring action to know the quality of the session. For this example, a quality degradation suddenly occurs and is detected. So, a new *Symptom* individual is generated in the ontology and SWRL engine is triggered again.

Now, *NotifySymptomAction* can be executed using rules similar to rules 1 and 2. "StreamingClientAgent" has not enough information to process this symptom, and it needs to cooperate with a Fully Autonomous Agent (in this case, the Fully Autonomous Agent that resides in the Multimedia Consumer Home Gateway, named "DiagnosisClientAgent"). "DiagnosisClientAgent" agent receives a message that notifies the new symptom. "StreamingClientAgent" receives an acknowledgement message and a new *SymptomACKMessage* individual is created and a postcondition is evaluated (see Rule 3. When all postconditions are satisfied, "sucessfullyPerformed" property is set to true.

Rule 3. SymptomACKMessage(?msg),
RequiredACKCondition(?condition),
PerformedAction(?action),
NotifySymptomAction(?action),
hasPostcondition(?action, ?condition),
hasSymptomContent(?msg,?symptom),
hasSymptom(?action, ?symptom),
→ satisfied(?condition, true)

"DiagnosisClientAgent" is able to process symptoms performing Bayesian inference in a distributed way (using MSBN approach). In other words, this agent shares information with others Fully Autonomous Agents that are able to reason with high level data. So, the received *Symptom* individual is translated to Bayesian format through the Mapping module (see Sect. 3).

Once this information is inserted into Bayesian module (see Sect. 3), all *Fully Autonomous Agents* are working together and in parallel thanks to MSBN. Each one takes its own decisions using all available knowledge (shared knowledge and its own private knowledge).

When "DiagnosisClientAgent" has processed the new symptom and a set of hypotheses are obtained from Bayesian module, it has to decide which action is the best to be executed now. Depending on the state of the environment and the knowledge base of the agent, one action could change its influence in the diagnosis process. To deal with this issue, we use the CDF method (see Section 3). With this method, all possible actions are ordered by relevance to reach a reliable confidence in the diagnosis process. The first one whose preconditions are fulfilled

is selected and executed. All Fully Autonomous Agents deployed in the scenario have to sort all possible actions to choose the best one.

For example, "DiagnosisServerAgent" has received shared knowledge when "DiagnosisClientAgent" has added the new symptom to the Bayesian module. So, "DiagnosisServerAgent" starts its own decision process to choose the best action that can be performed by itself. After it gets a set of *Hypothesis* individuals from Bayesian Module, then this agent chooses a *ConnectivityTestAction* to perform using CDF method (see Sect. 3). But it must check if all preconditions for this action are satisfied. In this case, two preconditions must be satisfied. One *RequiredActorCondition* (Rule 1) and one *RequiredDataCondition* (Rule 4).

Rule 4. RequiredDataCondition(?condition),
Variable(?variable),
requiredVariableType(?condition, ?typecond),
variableType(?variable, ?typecond)
→ satisfied(?condition, true)

Rule 4 searches all individuals of class *Variable* and class *RequiredDataCondition*. Then, it searches type of the condition variables. If a variable has the wanted type, then the condition is satisfied. In the case of *ConnectivityTestAction* the required variable type is "Streaming Server LAN IP" and the variable value is "192.168.1.11".

Since both preconditions are satisfied, the agent executes the test and obtains more information about the environment. The postcondition of this action is a *RequiredDataCondition* which checks if a result is obtained later the test is executed. This result is added to Bayesian module as evidence. The output of Bayesian reasoning process is processed by Mapping module and inserted in Ontology module as a new set of *Hypothesis* individuals. Once all hypotheses are updated, CDF method is executed again to get which action is the most relevant to be executed in order to reach a fast and reliable diagnosis. Then, the most relevant available action is executed and the making decision process starts again, i.e., to sort possible actions by relevance, to check preconditions, to filter available actions, to choose the most relevant available action and to execute it.

This process is repeated until, at least, one hypothesis has enough confidence, i.e. the confidence is higher than a threshold. Diagnosis conclusions are shared using MSBN approach and, then, diagnosis finishes and a healing action is searched to fix the problem if it is possible, otherwise the system notifies a human operator.

To summarise the case study explanation, several agents have been deployed in different devices to perform distribute diagnosis and the proposed upper-ontology has been applied (see Fig. 7). Some agents have all modules proposed in section 3. These agents can work isolated without problems and offer more functionality to our diagnosis system. But, there are some devices such as *dedicated multimedia server* that have low performance computational resources and it is not possible to deploy Fully Autonomous Agents in these devices. On one hand, to solve this problem, we deploy only key modules in several agents to

reduce required resources. But, on the other hand, an agent without Bayesian module cannot reach any diagnosis, because it only can perform actions (diagnosis tests), it does not perform inference. So, this *light-weight* agents have to share knowledge with Bayesian agents to reach diagnosis conclusions.

5 Evaluation

The benefits of the proposed upper-ontology have been evaluated comparing this approach with previous works [7,11]. In this paper, we compare the performance of the system using deliberation driven by "cost" or by "influence".

In previous works, test actions were ordered by estimated cost. This cost combined time cost and computational cost and it was estimated a priori by human experts. Then, all test actions are executed always in the same order. Even, sometimes, unneeded actions are executed.

In this work, test actions are sorted by relevance. Depending on the evidences about the environment, an action can modify its relevance for the current diagnosis (see Sect. 3).

The evaluation has been carried out based on a benchmark for a real diagnosis scenario of the R&D project Magneto [1]. With data stored in database with old diagnoses and the same Bayesian networks have been used in both cases. The volume of this data is around 500 diagnoses. We have clustered diagnoses in 13 diagnosis cases to simplify comparison and shown results.

As it is shown in Fig. 8, the number of performed tests has been reduced. Taking data from data base mentioned above, the average of performed test with deliberation driven by cost is 5.23 tests (with standard deviation 3.11). Using deliberation driven by influence, this number is reduced to 2.76 (with standard deviation 1.42); in other words, the number of performed tests has been reduced in 47.05%.

With deliberation driven by influence, there are two diagnosis cases that perform one test more than following the previous approach (driven by cost). The reason of this behaviour is that these are connectivity failures inside user HAN. These failures are very uncommon; for this reason, these hypotheses have, a priori, low confidence and other hypotheses have to be confirmed or refused first.

Table 1 shows the evaluation results in several columns. Mean Time to Diagnose (MTTD) [6] stands for the average time until the root cause of the failure is correctly diagnosed.

The column named "Result" represents if deliberation driven by influence improves driven by cost one or not in a specific diagnosis case.

The average of MTTD in previous approach is 25.47 seconds (with standard deviation 15.33), in proposed approach is 12.01 seconds (with standard deviation 7.12). Time improvement is 52.87%. These results show that the use of CDF method to extract the relevance of an action from Bayesian networks combained with semantic reasoning improves the performance of previous approaches driven by cost.

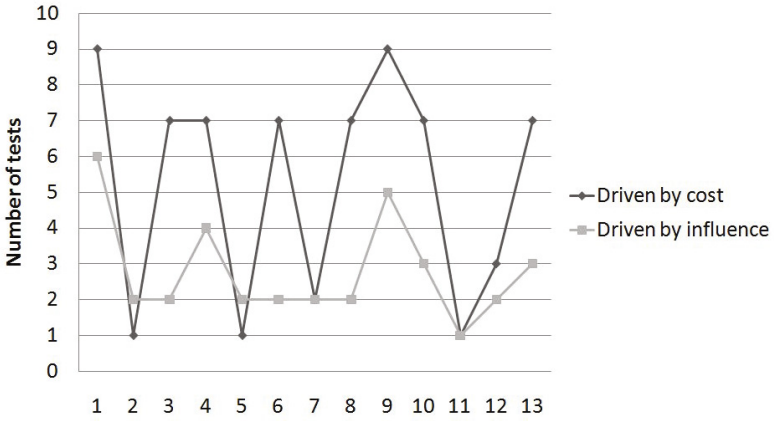

Fig. 8. Comparison: previous work vs proposed approach

Table 1. Evaluation of MTTD and number of tests using cost or influence metrics for ordering tests

Diagnosis case	MTTD		Number of tests		Result
	Cost	Influence	Cost	Influence	
Case 1	41	29	9	6	△
Case 2	5	7	1	2	▽
Case 3	36	8	7	2	△
Case 4	39	15	7	4	△
Case 5	4	8	1	2	▽
Case 6	36	8	7	2	△
Case 7	9	9	2	2	∼
Case 8	34	8	7	2	△
Case 9	40	24	9	5	△
Case 10	33	13	7	3	△
Case 11	5	5	1	1	∼
Case 12	13	8	3	2	△
Case 13	36	14	7	3	△

6 Conclusions and Future Work

We have presented a MAS that uses a diagnosis upper-ontology with Bayesian reasoning using OWL and SWRL to choose actions to perform. We focused on the deliberation process for hypotheses generation and discrimination. Our proposal of decision support improves previous approaches [7,11] both in time and in computational cost. Furthermore, the proposed modular architecture presents the capability to add or remove some modules of the architecture to reduce the resources required by agents, because there are important computational restrictions in devices like routers, TVs or STBs. Finally, one of the main advantages

of use Bayesian Networks and semantic reasoning is the possibility of deploy new knowledge base or new reasoning rules on the fly, without restarting or deploying new agents.

As future work, we will study in depth the application of MSBNs [12,13] to distribute the Bayesian inference engine that offers support to self-organisation capabilities to add robustness and to maintain coherence and consistency in a distributed reasoning process. For depth comparisons, we plan to simulate more complex network environment and to test several agent architectures in order to measure the performance of each architecture.

Acknowledgement. This research has been partly funded by the Spanish Ministry of Science and Innovation through the projects Ingenio Consolider2010 AT Agreement Technologies (CSD2007-0022), Cenit THOFU (CEN-20101019) and CALISTA (TEC2012-32457) as well as the Spanish Ministry of Industry, Tourism and Trade through the project RESULTA (TSI-020301-2009-31).

The authors would also like to thank to J. García-Algarra, J. González-Ordás, P. Arozarena and R. Toribio for their support and collaboration in this research through the *Magneto R&D project* [1].

References

1. Arozarena, P., Toribio, R., Kielthy, J., Quinn, K., Zach, M.: Probabilistic Fault Diagnosis in the MAGNETO Autonomic Control Loop. In: Stiller, B., De Turck, F. (eds.) AIMS 2010. LNCS, vol. 6155, pp. 102–105. Springer, Heidelberg (2010)
2. Benjamins, R.: Problem-solving methods for diagnosis and their role. International Journal of Expert Systems: Research and Applications 8(2), 93–120 (1995)
3. Berthet, G., Fischer, N.: A unified theory of fault diagnosis and distributed fault management in communication networks. In: Proceedings of IEEE AFRICON 1996, pp. 776–781. IEEE (1995)
4. Carrera, A., Iglesias, C.A.: B2DI - A Bayesian BDI Agent Model with Causal Belief Updating based on MSBN. In: Proceedings of the 4th International Conference on Agents and Artificial Intelligence, pp. 343–346 (2012)
5. Costa, P.C.G., Laskey, K.B.: PR-OWL: A framework for probabilistic ontologies. In: Proceedings of the 2006 Fourth International Conference on Formal Ontology in Information Systems, FOIS 2006, pp. 237–249. IOS Press (2006)
6. FitzGerald, J., Dennis, A.: Business Data Communications and Networking. John Wiley and Sons (2008)
7. García-Algarra, F.J., Arozarena-Llopis, P., García-Gómez, S., Carrera-Barroso, A.: A lightweight approach to distributed network diagnosis under uncertainty. In: INCOS 2009: Proceedings of the 2009 International Conference on Intelligent Networking and Collaborative Systems, pp. 74–80. IEEE Computer Society, Washington, DC (2009)
8. Kjaerulff, U.B., Madsen, A.L.: Bayesian Networks and Influence Diagrams. Information Science and Statistics. Springer, New York (2008)
9. Kraaijeveld, P., Druzdzel, M., Onisko, A., Wasyluk, H.: Genierate: An interactive generator of diagnostic bayesian network models. In: Proc. 16th Int. Workshop Principles Diagnosis, pp. 175–180. Citeseer (2005)

10. O'Connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., Musen, M.: Supporting Rule System Interoperability on the Semantic Web with SWRL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 974–986. Springer, Heidelberg (2005)
11. Sedano-Frade, A., González-Ordás, J., Arozarena-Llopis, P., García-Gómez, S., Carrera-Barroso, A.: Distributed Bayesian Diagnosis for Telecommunication Networks. In: Demazeau, Y., Dignum, F., Corchado, J.M., Pérez, J.B. (eds.) Advances in PAAMS. AISC, vol. 70, pp. 231–240. Springer, Heidelberg (2010)
12. Xiang, Y.: Belief updating in multiply sectioned Bayesian networks without repeated local propagations. International Journal of Approximate Reasoning 23(1), 1–21 (2000)
13. Xiang, Y., Poole, D., Beddoes, M.P.: Multiply Sectioned Bayesian Networks and Junction Forests for Large Knowledge-based Systems. Computational Intelligence 9(2), 171–220 (1993)
14. Wang, X.H., Zhang, D.Q.: Ontology based context modeling and reasoning using OWL. IEEE (March 2004)