A Real-life Application of Multi-Agent Systems for Fault Diagnosis in the Provision of an Internet Business Service

Álvaro Carrera[a], Carlos A. Iglesias[a], Javier García-Algarra[b], Dušan Kolařík[c]

[a] *Dept. of Telematic Engineering Systems, Universidad Politécnica de Madrid, ETSI Telecomunicación, Av. Complutense 30, 28040 Madrid, Spain*
[b] *Telefónica Investigación y Desarrollo, Ronda de la Comunicación s/n, 28050 Madrid, Spain*
[c] *IP network and services, Telefónica Czech Republic, Olšanská 6, 130 00, Praha, Czech Republic*

Abstract

Given that telecommunications networks are constantly growing in complexity and heterogeneity, management systems have to work with incomplete data, handle uncertain situations and deal with dynamic environments. In addition, the high competitiveness in the telecommunications market requires cost cutting and customer retention by providing reliable systems. Thus, improving fault diagnosis systems and reducing the mean time to repair with automatic systems is an important area of research for telecommunications companies. This article presents a Fault Diagnosis Multi-Agent System (MAS) applied for the management of a business service of Telefónica Czech Republic. The proposed MAS is based on an extended Belief-Desire-Intention (BDI) model that combines heterogeneous reasoning processes, ontology-based reasoning and Bayesian reasoning. This hybrid diagnostic technique is described in detail in the article. The system has been evaluated with data collected during one and a half years of system operation on a live network. The main benefits of the system have been a significant reduction in both the average incident solution time and the mean diagnosis time.

Keywords:

Email addresses: a.carrera@dit.upm.es (Álvaro Carrera), cif@dit.upm.es (Carlos A. Iglesias), algarra@tid.es (Javier García-Algarra), dusan.kolarik@o2.com (Dušan Kolařík)

network management, agent, Bayesian network, diagnosis, uncertainty, BDI

1. Introduction

Fault Diagnosis is one of the most important network management tasks for telecommunications companies. Traditionally, this process has been carried out by humans and software systems that work in a cooperative way. The constant increase in the size and complexity of the network, makes fault diagnosis a critical task that should be handled quickly and in a reliable way. High skilled engineers are required to carry out this task, although even these individuals are not always able to deal with the increasing heterogeneity and complexity of the networks. These fault diagnosis tasks require the processing of uncertain knowledge from geographically distributed devices or systems from different vendors with different interfaces or protocols, making data collection and understanding difficult. Although automated fault diagnosis processes have been developed, such as surveillance systems for symptom detection in the core or backbone networks, fault diagnosis is mainly a manual process managed by human operators.

Nevertheless, operators have the goal of fully automating fault diagnosis to reduce operation costs and improve customers' experiences through the automated operation of standardised diagnostic processes. Moreover, the increasing heterogeneity of networks makes it difficult to cope with their complexity, necessitating an automated approach.

This article presents a Multi-Agent System (MAS) architecture designed to assist human operators in diagnosis of network faults. The proposed MAS architecture is based on an extended BDI model that combines Bayesian reasoning to handle the uncertainty with ontology-based reasoning for domain knowledge inference. This system has been applied to a service of Telefónica Czech Republic. The system has been evaluated based on data collected during one and a half years of operation. The results show that the system is being widely used by human operators and that the incident solution time has been reduced.

The rest of this article is structured as follows. First, Section 2 provides an overview of the MAS design following the Prometheus methodology [1]. Then, Section 3 describes the scenario in which the system has been deployed. Section 4 presents the results of implementation of the system in a real scenario. Finally, we discuss related works in Section 5 and Section 6 provides conclusions and discusses potential future work.

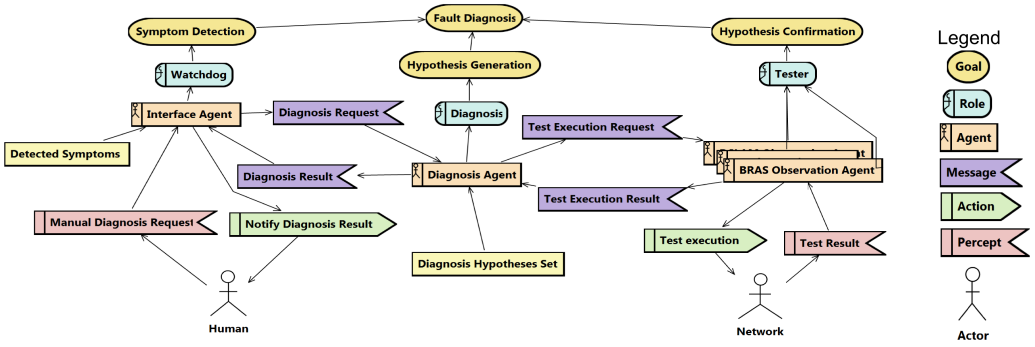

Figure 1: Analysis Overview Diagram for a Fault Diagnosis Scenario.

2. Design of a MAS for Fault Diagnosis

To cope with the complex requirements of users in the telecommunications industry, an agile methodology has been defined, BEhavioural Agent Simple Testing (BEAST) based on Behaviour Driven Development (BDD). This methodology is focused on the iterative validation of user requirements with executable acceptance tests as detailed in [2]. This section describes the design and analysis of the MAS. For this purpose, we use the well known methodology Promotheus [1], which is compatible with BEAST, although other Agent Oriented Software Engineering (AOSE) methodologies such INGENIAS [3] or MASE [4] could have been selected. The Prometheus methodology supports the development of intelligent agents by non-expert users in a practical, leading to its application in both industry and academia. Prometheus consists of three design phases that are explained in the following subsections: system specification(Section 2.1), the architectural design phase (Section 2.2) and the detailed design phase (Section 2.3). The following images depicting the system design were created with the Prometheus Design Tool (PDT) [5].

Fig. 1 presents an overview diagram of the developed system, indicating the goals of the different roles and the interactions of the system with external actors (human operators and network devices).

2.1. System Specification

This phase provides a general overview of the system, specifying the global goals (Section 2.1.1) and a set of roles identified (Section 2.1.2) for the fault diagnosis scenario.

2.1.1. Scenario and Goals

Based on Benjamins' model of diagnosis [6], there are three well-distinguished steps in any diagnosis scenario: (a) *symptom detection*, (b) *hypothesis generation* and (c) *hypothesis confirmation* (see Fig. 1). Each time a new symptom is found, a set of hypotheses is generated. This set contains possible root causes of the fault based on the symptoms. Then, each hypothesis must be confirmed or discarded to obtain reliable conclusions. Tests are executed to determine the state of the system and the results are fed back to the hypothesis generation step in order to update the current set of hypotheses with the available knowledge.

2.1.2. System Roles

The subtasks identified in the model (see Fig. 1) must be carried out by agents in the automated fault diagnosis system. Based on these tasks, a set of roles has been modelled for the proposed MAS: *Watchdog*, *Diagnosis* and *Tester*.

The *Watchdog* role is responsible for detecting new symptoms and starting the diagnosis process. This type of agent must be specialised to interact with its environment, providing integration with devices and/or services. For example, an agent that plays the *Watchdog* role should recognise if a Real Time Streaming Protocol (RTSP) session has poor quality (i.e. issues such as packet loss or jitter). Thus, the *Watchdog* role must be integrated with network management systems.

The *Diagnosis* role is responsible for generating plausible hypotheses (fault root causes) based on symptoms and test results. This role chooses which tests must be performed to confirm hypotheses. In addition, test scheduling and prioritisation can also be defined. This agent should also be tailored for a particular diagnosis domain, such as a specific network region such as the outer edge or technology such as Fiber To The Home (FTTH) or a provisioning subsystem.

The *Tester* role is responsible for performing tests on demand to obatin updated information about the environment, including device or service states and configurations. As for the rest of the roles, this role should also be specialised for interaction with its environment. For example, an agent that plays the *Tester* role could be required to perform a test to determine the CPU load of a server.

The main difference between the *Watchdog* and *Tester* roles is the proactive nature of the *Watchdog* role. The *Watchdog* is continuously monitoring

the status of a set of variables to achieve its goal of detecting new fault symptoms. In contrast, the *Tester* role performs tests on demand.

These roles have been defined to enable the propose multi-agent architecture to be deployed in a flexible progressive system, as some roles can be played by either human operator or software agents.

Note that one single agent can play one or several roles depending on the available resources.

The proposed roles have been defined as simply as possible so that agents at critical nodes can carry out the minimum required tasks, while nodes without resource consumption restrictions can employ fully functional agents. Following this approach, a *Diagnosis* agent can play both *Diagnosis* and *Tester* roles if these tests are performed in the same node, while in a lightweight device such as a user home gateway, a *Tester* agent must play only this role due to computational restrictions.

Furthermore, several advanced roles can be modelled to offer enriched fully autonomous fault diagnosis. Other interesting features, including self-learning, self-healing, self-configuration and self-adaptation, may be added to the model using other roles. For example, *Healer* and *Worker* roles can be used to add self-healing capabilities using the output of a *Diagnosis* agent. The *Healer* role is responsible for choosing healing actions to restore the diagnosed system to the correct behaviour, while the *Worker* role performs actions chosen by *Healer* agents. For example, a *Worker* agent could execute an action to close a specific port in a firewall. To add self-learning capabilities, *Reporter* and *Trainer* roles could be added to the system. The *Reporter* role is responsible for determining whether a healing action achieved the expected result i.e., to check whether the symptoms have disappeared and the status of the diagnosed system has been properly restored. Finally, the *Trainer* role is responsible for machine learning based on previous diagnoses, to improve the hypothesis generation and symptom detection phases.

2.2. Architectural Design Phase

This phase defines the interactions among the agents in the system based on the roles defined in the previous phase (Section 2.1).

2.2.1. Agent-Role Grouping

The grouping of roles to agents is straightforward (see Fig. 1). The *Watchdog* role is played by the *Interface agent*, which receives symptoms and starts the process of diagnosis. The *Diagnosis* role is directly associated with the

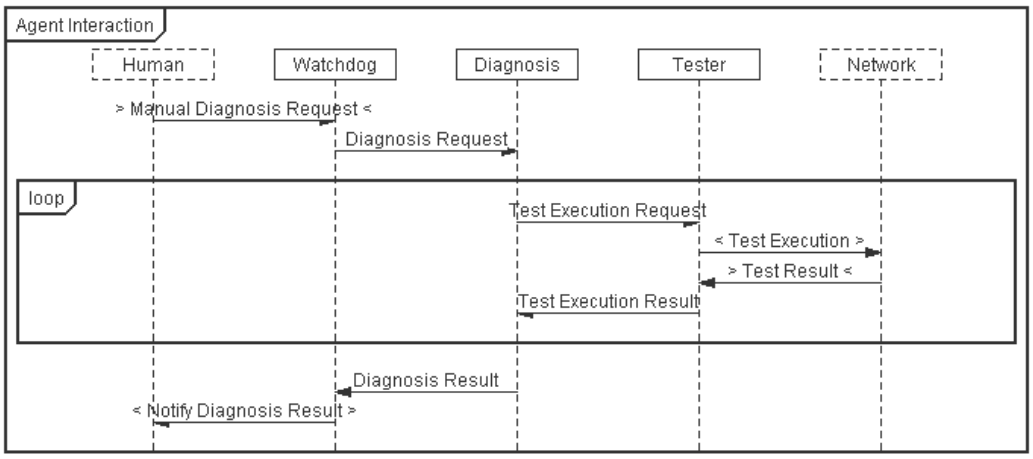

Figure 2: Interactions among agent roles.

Diagnosis agent. Finally, the *Tester* role is played by a set of *Observation agents* that are specialised to test for different device types.

2.2.2. Message flow

The interaction flow in a standard diagnostic process starts when a *Watchdog* agent detects a failure symptom. This symptom is sent to a *Diagnosis* agent, which then generates a set of hypotheses and requests several specialised *Tester* agents to perform tests. Once enough information about the environment is available to reason about the situation and the hypotheses have been confirmed or discarded, the diagnostic process ends. This process is shown in Fig. 2, where one human actor has been included. In this case, the human actor reports a new symptom and receives the results of the diagnosis.

2.3. Detailed Design Phase

This phase details the internal design of the proposed agent architecture for a diagnosis scenario. Uncertainty is one of the main challenges facing a diagnostic process. At the beginning of the diagnostic process, only partial information is known, and it is not feasible to collect or even model all the available information in complex environments such as telecommunications networks that are composed of multiple subnetworks and equipment in constant evolution. Thus, diagnosing telecommunications networks requires reasoning under uncertainty.

Reasoning Technique	Rules systems	CBR	Fuzzy logic	Bayesian inference
Coherence/Consistency	Good	Good	Bad	Good
Handle uncertainty	Null	Null	Good	Good
Failures tolerance	Medium	Bad	Medium	Medium
Maintain private data	Good	Medium	Good	Medium
Uncompleted data set	Bad	Good	Medium	Good

Table 1: Comparison among reasoning techniques [8].

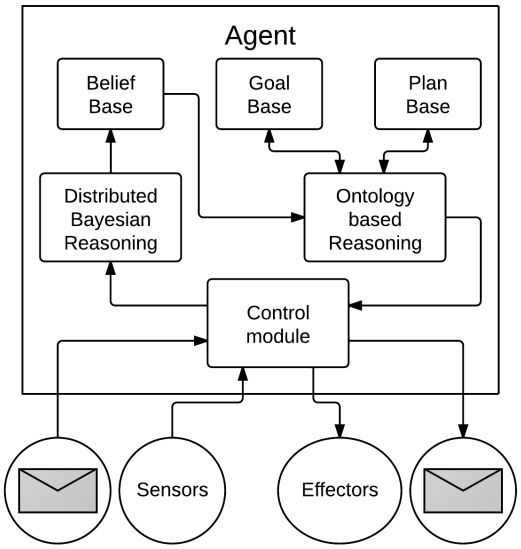

Figure 3: Extended BDI Model.

The reasoning technique chosen to meet these requirements is the probabilistic reasoning offered by Bayesian Networks (BNs) in a distributed way (Multiply Sectioned Bayesian Networks (MSBNs) [7]). Table 1 presents a comparison of reasoning techniques in accordance with Zhang [8]. Compared with other alternatives like rule-based or case-based reasoning, Bayesian inference performs better on incomplete data sets and in the presence of uncertainty while maintaining coherence and consistency. Zhang presented a good review and comparison of these reasoning techniques [8]. Thus, agents that handle uncertainty use a Causal Model CM, i.e. a BN. This BN is used to perform distributed causal inference to update beliefs. The CM is a subnet that is part of the MSBN [7], allowing to maintain coherence during the distributed reasoning process.

An extended BDI model has been designed to meet the requirements

discussed above such as coherent distributed reasoning and handle incomplete datasets. In this model, distributed Bayesian reasoning is used to handle uncertainty in a scalable way. The overall causal model is partitioned across the multi-agent system, based on its domain specialisation and geographical distribution in the network. Thus, agents can generate beliefs and/or goals based on the perceived environmental data (e.g., alarms of new faults or monitoring processes). Agents with the *Diagnosis* role use the distributed causal model based on MSBN to determine the root cause and are able to reason about potential repair actions using a specialised ontology.

Given that the classical BDI architecture does not include probabilistic reasoning, the BDI model has been extended to deal with uncertainty. Moreover, we have extended the model to reason with *Distributed Bayesian Reasoning* because the findings in a particular domain (or region) can be causally connected to the findings in another domain (or region). Fig. 3 shows the agent architecture and Alg. 1 presents the extended control loop of the proposed extended BDI model.

The proposed architecture is composed of three main modules: *Control Module* (see Section 2.3.1), *Distributed Bayesian Reasoning Module* (see Section 2.3.2) and *Ontology-based Reasoning Module* (see Section 2.3.3).

2.3.1. Agent Control Loop

The standard BDI control loop [9] has been extended to provide coherence and consistency in the causal distributed reasoning. The *Control Module* executes the control loop and interacts with the rest of the modules. In the proposed extended control loop (Alg. 1), an agent begins with initial beliefs and intentions. These beliefs are added to the CM and synchronised with other agents through the MSBN. In this algorithm, the CM variable denotes the Causal Model used by an agent during its reasoning process and B, D and I stand for agent Beliefs, Desires and Intentions, respectively.

This algorithm uses several functions. This control loop is similar to the classic BDI model, but it has been extended with the following functions: *connect* function, *belief initialisation* function, *update notification* function, *belief update* function and *communicate belief* function.

The proposed algorithm starts with the initialisation of the beliefs and intentions of the agent. B_0 and I_0 represent the Initial Beliefs and Initial Intentions as in a standard BDI control loop [9]. Then, the *connect* function is used to set-up the agent CM with the distributed Bayesian inference engine. Once the CM is ready, the *belief initialisation* function is called to synchro-

nise the shared knowledge among all agents. When beliefs are initialised and synchronised, the control loop starts. Now, the agent waits until a perception or a notification is received. If the agent receives a notification of belief update from the *update notification* function of another agent, it updates its own beliefs through a local inference process (*belief update* function). If the agent receives a new perception, it updates its own beliefs using the *belief revision function (brf)* function as in a standard BDI control loop [9].

After this, the agent propagates its updated beliefs to other agents using the *communicate belief* function. This function simply notifies agents when a change occurs in the set of shared beliefs.

Finally, the standard BDI functions (*options*, *filter* and *plan*)are performed by the Ontology-based reasoning module to choose which plans must be executed. This control loop ends as follows. A set of possible plans are chosen (*options* function) among all available plans and the best plan to achieve the current goals is selected (*filter* function) and executed (*plan* function). After this, the agent waits again until a perception or a notification is received.

The proposed algorithm (Alg. 1) is robust because agents can join or leave the MAS using *update notification* and *belief update* functions. Upon receiving a notification that a new agent is operating or that an old agent has left the system, agents start the reconfiguration process to maintain coherence in the MSBN (for more details, please refer to the work of Xiang [7]).

A mapping process has been defined to communicate between the Bayesian and Ontology reasoning modules (Fig. 3). This mapping process extracts ontology individuals from the Bayesian network and populates them in the ontology module.

The details of the reasoning processes in the Bayesian and Ontology modules are described in the following subsections, Section 2.3.2 and 2.3.3, respectively.

2.3.2. Causal Modelling and Reasoning

As described previously, we propose to use Bayesian reasoning to cope with uncertainty in the process of diagnosis. The Bayesian network design follows the BN3M [10] model, which classifies network nodes (variables) into three groups: *evidence*, *root causes* and *context* (including *auxiliary variables*). Once the variables of one domain are defined, the next step is defining their connections through a Conditional Probability Table (CPT).

These probabilistic relationships (domain knowledge) can be obtained

Algorithm 1 Extended BDI Agent Control Loop

1: $B := B_0$;
2: $I := I_0$;
3: $CM := connect(CM)$;
4: $B := belief\ initialisation(B, CM)$
5: **while** *true* **do**
6: **if** $update\ notification(CM)$ **then**
7: $B := belief\ update(B, CM)$;
8: **else**
9: *get next percept* ρ;
10: $B := brf(B, \rho)$;
11: **end if**
12: $CM := communicate\ belief(B, CM)$;
13: $D := options(B, I, CM)$;
14: $I := filter(B, D, I, CM)$;
15: $\pi := plan(B, I)$;
16: $execute(\pi)$;
17: **end while**

from human experts, based on data-mining techniques applied to historical data or based on a mixed process.

During a diagnostic process, the causal model is used in the hypothesis generation step. Available information about the diagnosis scenario is added progressively to the BN (symptoms, tests results, context variables, etc.). These data are managed as *evidence*, i.e., beliefs with 100% confidence.

Each time a new piece of evidence becomes available, a set of hypotheses with an associated degree of confidence is generated by the Bayesian Reasoning Module. The MSBN reasoning technique allows for several agents to reason concurrently with their own local Bayesian Network and then distributing their findings to the other Bayesian networks. This requires causal models to be properly connected as described in Alg. 1. Some variables of the Bayesian Networks behave as bridges for connecting the local Bayesian Network, and an initialisation and update algorithm is used to maintain coherence among the subnets, as detailed by Xiang et al. [7].

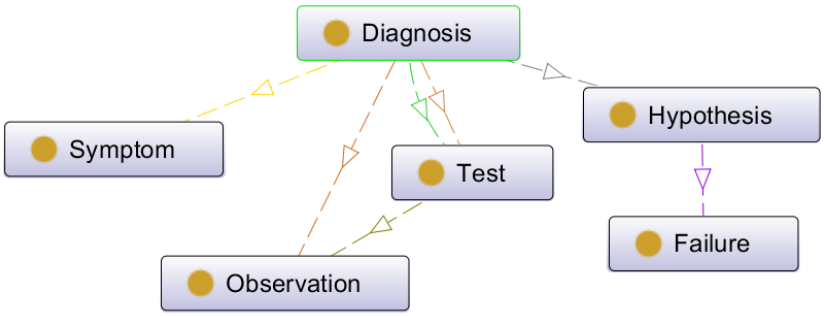

Figure 4: Diagnosis Ontology.

2.3.3. Ontology Modelling and Reasoning

An ontology has been developed to reason about the outcomes of the Bayesian Reasoning Module. Thus, agents can reason about the best strategy to repair a diagnosed fault based on their local knowledge of the potential repair actions. This ontology, as shown in Fig. 4, is based on the following main concepts (highlighted in italics). A diagnosis is composed of a body of evidence (*symptoms* and test results, referred to as *observations*) and *hypotheses* of *failure*. A set of *tests* can be carried out to obtain information about the environment and feed that information back to the hypothesis generation step.

Depending on the available information, the Ontology-based Reasoning Module chooses a test to execute to achieve a rapid and reliable diagnosis.

The acquisition of the knowledge from the causal model is performed to create individuals of the ontology following the steps outlined below. Each time a diagnostic process starts, the individuals for this ontology are extracted from the causal model. All nodes of the causal model can be classified into one of three types mentioned above for the BN3M [10] model: evidence, fault root causes and context. Initially, individuals representing *symptoms* are created based on the information received in the diagnosis request message (see Fig. 2) and individuals representing *hypotheses* are extracted from the causal model based on the nodes that represent fault root causes. Later, individuals representing *tests* are extracted to analyse which evidence can be used in the causal model. These *test* individuals are classified as possible tests for the current diagnosis. At this point, the hypothesis confirmation loop begins (see Fig. 2). Each time a new test result is received, a new *observation* individual is generated to represent the result, and the test is marked

as performed. Now, Bayesian inference is performed again and a new set of updated *hypotheses* are generated to replace the previous set. When all possible tests have been executed or one or several hypotheses reach a threshold, the diagnostic process is complete and the final hypotheses are sent to the agent that had requested the diagnosis.

To summarise, the design of the MAS presented in this section is based on a BDI architecture that has been extended to cope with uncertainty management. The proposed model combines the uncertain reasoning abilities of the Bayesian network with domain knowledge about the network equipment. This domain knowledge is modelled using the proposed ontology, which stores the information required to carry out the diagnosis tasks. The scalability of the solution is dealt with in two ways. First the solution is distributed thanks to the MAS architecture we have defined based on different agent roles. Then, Bayesian reasoning is distributed using MSBN. In conclusion, the solution proposed here is scalable and adaptable to the growing heterogeneous systems that must be managed by a telecommunication company.

3. Case study

This section presents practical experience in the application of the Fault Diagnosis MAS proposed in the previous sections. The system has been implemented at Telefónica O2 Czech Republic to manage a *Internet Business* service. This scenario is described in Section 3.1. Then, the agents deployed and the specific features developed to adapt the model to the client requirements are presented in Section 3.2.

3.1. Scenario

Internet Business is a service for business subscribers offered and operated by Telefónica O2 Czech Republic. The service provides secure Internet access to corporate users based on Virtual Private Network (VPN) technology. This system was selected for this study because it is easy to understand but provides an interesting scenario with enough complexity to evaluate the proposed diagnosis system. Fig. 5 depicts the technical infrastructure required to offer this service. In this infrastructure, subscribers communications equipment connections are realised via a multi Symmetric Digital Subscriber Line (SDSL) to a Digital Subscriber Line Access Multiplexer (DSLAM). Traffic from the DSLAM is transported through the Regional Ethernet Network (REN) to the entry point of the Multiprotocol La-

bel Switching (MPLS) network where MPLS pseudo-wire connections are established to the MPLS provider equipment at a central site. Finally, the customers communication equipment establishes a Point-to-Point Protocol (PPP) session with the Broadband Remote Access Servers (BRASs) at the central site using this transport path. The use of technology from different vendors for the different network elements increases the complexity of the diagnosis.

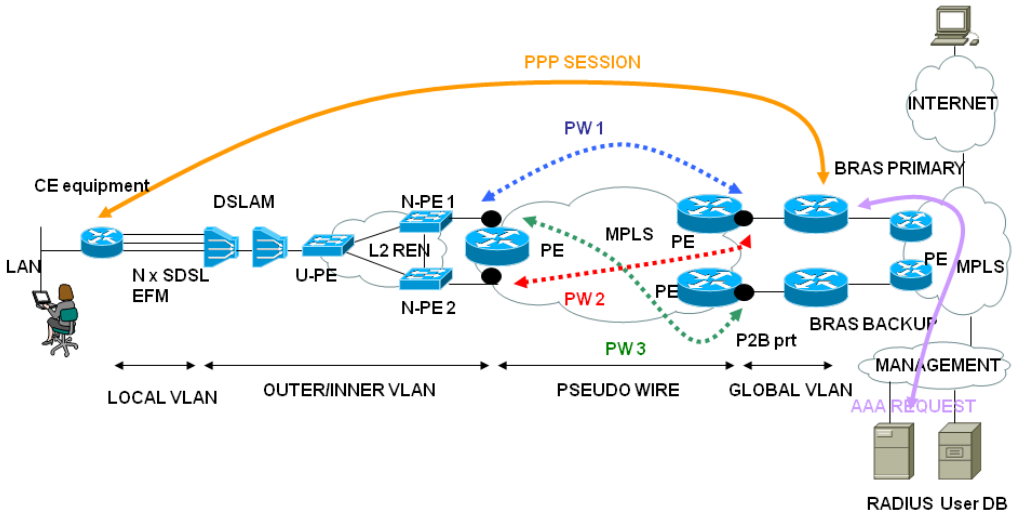

Figure 5: Technical infrastructure for providing the Internet Business service.

This comprehensive technical solution imposes strong requirements on the inventory and configuration systems.

One of the main causes of failure in this service is configuration issues of inventory systems. An inventory system allows for the precise identification of the network elements (physical or virtual, including their technical characteristics) that are being used to offer a service to a subscriber. The scenario includes a combination of automatic configuration systems based on even networks and on a human based configuration of the inventory system (e.g. assigning a new IP address or VLAN). When a configuration change request process is initiated, a service outage or a decrease in the quality of service could occur if the Operation Support Systems (OSS) or inventory systems fail. Other potential causes of service outages may be hardware or software failures or last mile problems. Hence, in summary, there are many possible root causes of failure in this scenario. The data required to carry

out a diagnosis are geographically distributed. Moreover, the information can be missing, outdated or even unreachable. Thus, this diagnosis scenario is suitable for the application of uncertainty reasoning techniques.

The main reason for using the proposed diagnosis system in Telefónica O2 Czech Republic is to decrease the Mean Time to Diagnose (MTTD) [11] . In addition, a more effective diagnosis system would also increase customer satisfaction and decrease the human resources required for diagnosis tasks. The Internet Business service was selected for automated diagnosis because of the high number of customers using this solution, the high number of trouble tickets and the high level of complexity of the service.

3.2. Fault Diagnosis MAS Deployment

Following the model presented in Section 1, a MAS based diagnosis system has been developed and deployed to the OSS servers. This system was developed using the JADE platform [12], which offers an open agent model that allowed us to build the proposed extended BDI model (see Fig. 3). For testing purposes, the MASON framework [13] has been used to simulate network devices and services.

Human operators are able to interact with the Fault Diagnosis MAS to request diagnoses using a web interface. In this request, they provide the detected symptom to an *Interface Agent* that is responsible for collecting data from inventory databases.

Once the required information has been collected, the *Interface Agent* sends a diagnosis request message to the *Diagnosis Agent*. Detected symptoms are added to the Bayesian inference engine, which handles the causal model of the fault, to generate a set of hypotheses that represents possible root causes of the fault. Later, the *Diagnosis Agent* requests *Tester* agents to perform tests.

There are six *Tester* agents deployed in this scenario, with each agent specialising in a device type: customer equipment, provider equipment, BRAS, DSLAM, REN and inventory databases. For this task, shell scripts and Simple Network Management Protocol (SNMP) commands are used to interact with network elements. Each time a test is performed, the result is fed back to the causal model and *Diagnosis Agent* generates new updated hypotheses.

If one or more hypotheses attain a sufficiently high level of confidence, the final hypothesis set is shown to the network operator who requested the diagnosis through the web interface. To make the diagnosis results easy to understand, the output of the diagnosis system is presented to the human

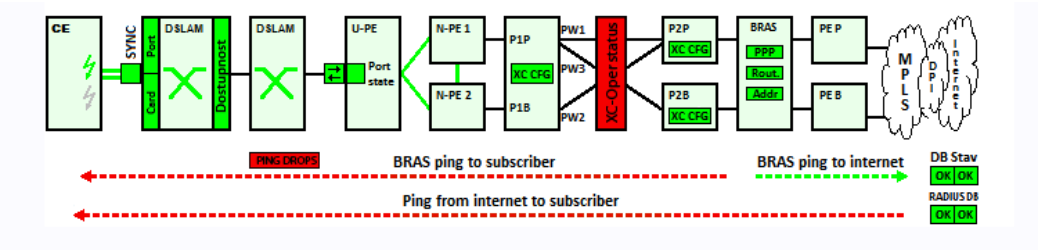

Figure 6: Diagnosis results graph.

operator in a diagram, as shown in Fig. 6. This diagram summarises all the information retrieved in the diagnostic process.

In this scenario, the *Diagnosis Agent* carries out the hypothesis generation and reasons under uncertainty as shown in Section 2.3.2. The Bayesian network that represents the fault causal model used by this agent is composed of 27 evidence nodes (i.e., symptoms and test results), 17 hypothesis nodes (i.e., possible root causes of the fault) and 18 auxiliary nodes (62 nodes in total). A portion of this causal model is presented to explain the diagnostic process used in the case study (see Fig. 7).We highlight the node that represents the detected symptoms (e.g., *Manifestation*), the three nodes that represent fault root causes hypotheses (shown with a blue background) and the six nodes that represent tests results (shown with a yellow background). The rest of the nodes are auxiliary nodes.

These nodes are translated into ontology individuals to reason with them, as was described in Section 2.3.3. Fig. 8 presents the result of the mapping process that generated individuals based on the causal model subnet shown in Fig. 7.

In summary, the proposed fault diagnosis model has been successfully applied to an *Internet Business* service. A web interface has been developed to allow human operators to interact with the diagnosis MAS through an *Interface Agent* that receives symptoms and collects information to start a diagnosis. The *Diagnosis Agent* performs Bayesian inference using a causal model that represents the relationhip between network status and possible failures of the *Internet Business* service. Six different *Observation Agents* perform tests using shell scripts in accordance with the demands of the *Diagnosis Agent*. Finally, the diagnosis result is shown to the human operator (see Fig. 6).

In the diagnosis system deployed here, the roles described in Section 2.1.2

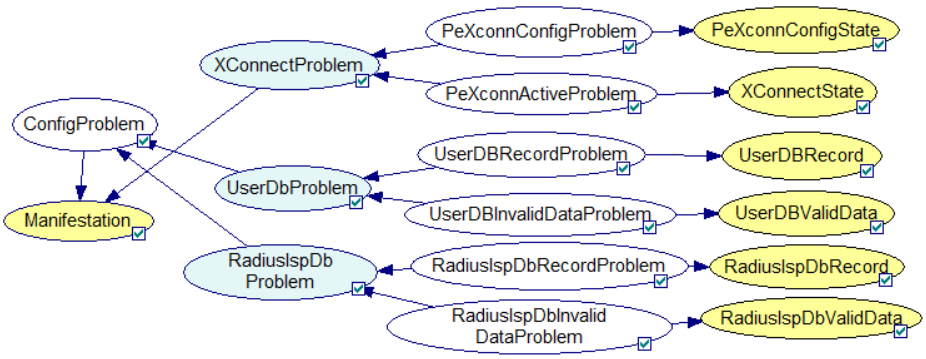

Figure 7: A portion of the causal model used in the case study. The associated CPT of each node is omitted.

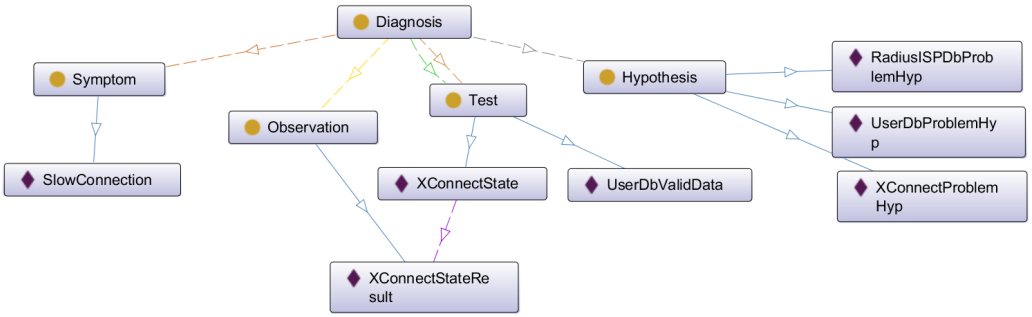

Figure 8: Example of Ontology Individuals obtained with the mapping process.

are played by both human and software agents. The symptom detection step is carried out by human operators who use a web page to request the MAS to perform a diagnosis. Then, an agent collects the required parameters to perform tests from the inventory system, and a set of tests are executed to retrieve information about the current status of the network. For this task, shell scripts and SNMP commands are used to interact with networks elements such as REN devices, DSLAMs and BRASs.

4. Evaluation

This system was evaluated during the one and a half year period from November 2010 to March 2012. During this time, the fault diagnosis MAS deployed in the scenario presented in Section 3 was operating and recording data on diagnosis cases stored in a database in internal Telefónica Czech

Republic servers. This database represents around 8600 different cases. Every diagnosis case contains information about which tests were performed, what information was available during the diagnosis process and what final conclusions were reached at the end of the diagnostic process.

The evaluation methodology consisted of two steps. First, we analysed the coverage of the dataset relative to the global problem to check whether these data are sufficiently representative as explained below. Then, we defined several Key Performance Indicators (KPIs) to evaluate the business benefit of the system.

To analyse the complexity of the scenario, the entropy of possible diagnosis cases has been calculated and compared with the entropy of each possible root cause. This entropy represents how a same fault root cause can be manifested in the environment for different test results. Many different test result sets can result from the same fault type. In this case, 17 different fault types have been identified. The Bayesian network has been successfully used to reason under this uncertainty. The normalised entropy of each possible root cause was compared to determine which fault root cause (fault type) is more complex. Fig. 9 shows that some root causes have entropy values close to zero, because these fault types almost always present the same symptoms. In contrast, other fault root causes exhibit high entropy because these fault types can be manifested as different symptoms and test results.

To graphically represent all diagnoses stored in the database, a Sammon mapping algorithm [14] has been used to represent the diagnoses in a two dimensional graph (see Fig. 10). Using this algorithm, the relative euclidean distance among all stored diagnoses is maintained. As shown in Fig. 10, diagnoses with the same highest percentage hypothesis (i.e., the same most probable cause of failure) are close to one another in the graph (Fig. 10). To highlight these clusters, each one has been rounded and labelled properly. Furthermore, to confirm that the Sammon mapping algorithm is a good way to represent the complexity of each fault root cause, the area of each region is directly related with its entropy (a high entropy is represented with a large region). To interpret this graph, note that two diagnosis cases that are graphically in the same place in Fig. 10 represent cases with the same symptoms and the same final hypotheses, i.e., the euclidean distance between these cases is zero.

The duration of diagnosis is presented in a histogram in Fig. 11. Note that Y axis of Fig. 11 is on a logarithmic scale. The result of this histogram can be understood as a Gaussian distribution with a mean value of 48.365

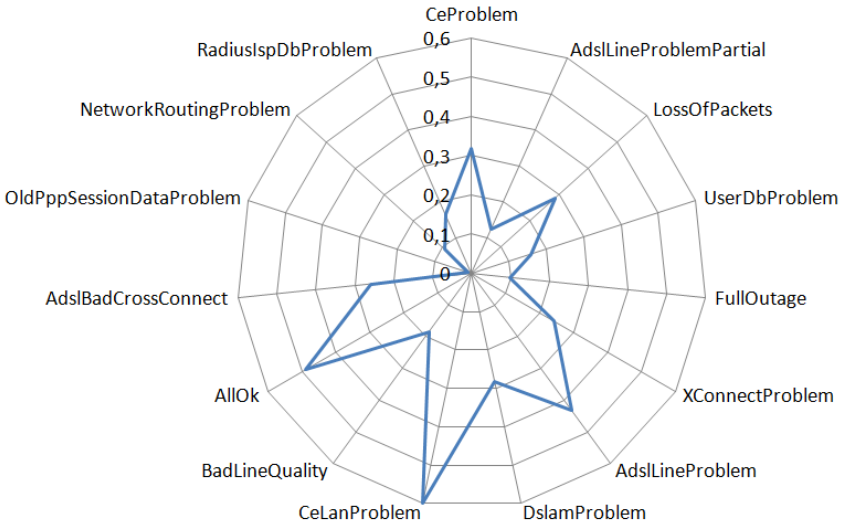

Figure 9: Normalised entropy of various root causes of faults.

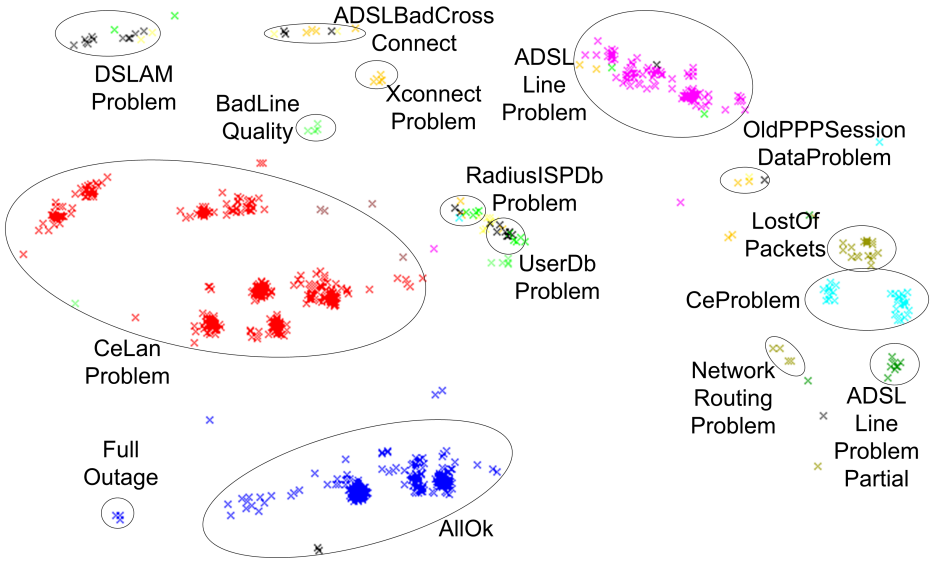

Figure 10: Fault root cause clusters.

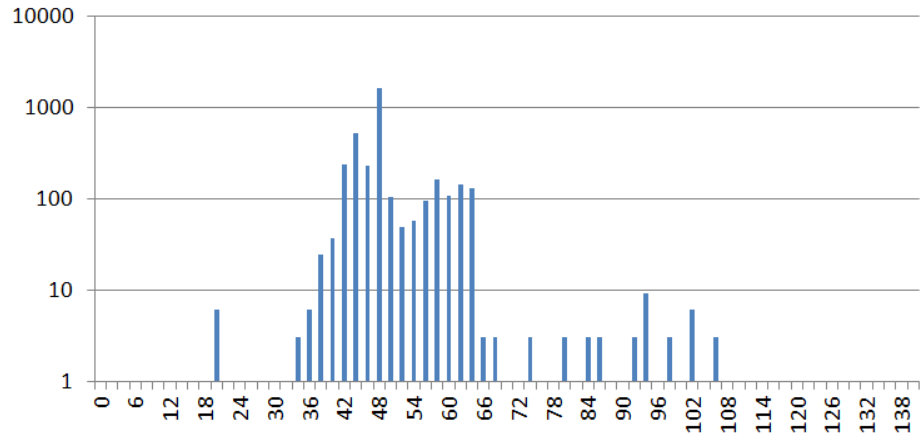

Figure 11: Histogram of diagnosis duration (in seconds).

seconds and a standard deviation of 7.462 seconds.

The main conclusion of this analysis is that the available database covers the majority of the possible diagnosis cases that can occur in the scenario under study, as the normalised entropy and the Sammon mapping graphs show that there are many variations of these diagnosis cases in the dataset. Thus, several KPIs have been defined (see Table 2) to evaluate the business benefits of the system. In summary, the KPIs values are meaningful as the available dataset is sufficiently representative relative to the global problem.

KPI1 is used to measure the usage of the system by human operators, i.e., the acceptance rate of the diagnosis system. This KPI was initially 24.74%, and after the introduction of the MAS diagnosis system, it increased to 92.00%. In other words, 92% of diagnosis cases use the solution presented by the system.

KPI2 is used to measure the average incident solution time (i.e., the diagnosis and repair time). Initially, this KPI was 9.51 hours, and the MAS, it has decreased this value to 5.2 hours, representing 45.32% time savings.

KPI3 is used to measure the mean time before a work order is created (i.e., the diagnosis time). This KPI was initially 0.56 hours, and the MAS system has decreased this metric to 0.17 hours, representing 33.93% time savings.

Table 2 collects the initial and final metrics. During this evaluation period, a human technical expert has periodically improved the causal model, i.e., the Bayesian network, based on historical data and on feedback from

KPIs	1st Month	18th Month
KPI1	24.74%	92.00%
KPI2	9.51 hours	5.2 hours
KPI3	0.56 hours	0.37 hours

Table 2: System KPIs

human operators using the fault diagnosis system.

5. Related Work

Both the size and complexity of telecommunications networks have increased exponentially in recent decades. However, methods of managing these networks and handling their faults have not kept pace. Thus, the diagnosis process is still an almost entirely manual process, and new alternatives are being explored. The use of MAS for network diagnosis has been previously discussed by several researchers [15, 16, 17]. Martín et al.[18] present a framework based on intelligent agents for network management that used rule-based reasoning. Telefónica has largely used rule-based systems, but rule-based maintenance has been an issue. The use of Bayesian networks has considerably reduced the required effort, and introduced capabilities of learning and uncertainty management. Both Leitão et al. [19] and Mendoza et al. [20] present MAS focused on the reconfigurability of the system for collaborative tasks using adaptable agents. In our evaluation, we have defined agents with specialised roles, although in future research we will evaluate whether the system is improved by allowing all the agents to play all the roles. Luo et al. [21] proposed a fault diagnosis system using Dempster-Shafer evidence theory [22, 23] with combination rules to resolve the possible conflicted information from multi-sensors systems. In our work, we solve this issue using Bayesian networks to handle the uncertainty of the incomplete or unreliable data.

6. Conclusions and Future Work

This article has proposed a Multi-Agent System (MAS) for fault diagnosis, and has been validated in a real scenario in the business internet service provided by Telefónica O2 Czech Republic.

The performance of the system has been measured with several KPIs (see Section 4) that demonstrate the acceptance of the new diagnosis system by human operators and the reduction in time of the incident solution time. Furthermore, the diagnosis results stored in a database during the one and a half year period of operation of the diagnosis MAS have been analysed to measure the entropy of all available diagnoses (see Fig. 9) and to graphically represent the similarity among the diagnoses (see Fig. 10).

An important conclusion of this research work is that agent technology is suitable for distributed diagnosis. Agent technology has proven to be very useful for adapting the identified roles to different domains without requiring extensive training.

The proposed MAS architecture is sufficiently flexible to enable progressive deployment of agents to replace human-based tasks. This strategy has also be a key in the success of the deployment of the system.

To ensure scalability, the MSBN approach offers the ability to use local causal models (BNs) that can work together. This feature allows for the Bayesian knowledge to be split into several agents that can dynamically reconfigure their causal reasoning models dynamically, depending on changes to the environment. From a different point of view, the ability to split the causal model facilitates the design, maintenance and reuse of Bayesian networks. Furthermore, the MSBN approach has been followed in the case study for Telefónica O2 Czech Republic to allow for knowledge sharing (test results, hypothesis sets, environmental data, etc.) to maintain coherence during the Bayesian reasoning process among agents that diagnose different services.

Now that the architecture has been validated in the scenario presented here, several possible paths can be explored. To enable rapid development and validation cycle, we are working in a simulation environment based on the MASON [13] environment to simulate advanced information retrieval from the network. Since the MAS works in a simulated environment, we can experiment with self-healing and self-configuration capabilities in ways that are not possible ni real environments due to the possibility of misconfiguring any device that is assigned to human operators. Furthermore, we plan to develop an advanced knowledge management agent that will be able to deploy agents on demand depending on the requirements of each diagnostic process.

References

[1] L. Padgham, M. Winikoff, Prometheus: A methodology for developing intelligent agents, in: F. Giunchiglia, J. Odell, G. Weiss (Eds.), Agent-Oriented Software Engineering III, Vol. 2585 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 174–185.

[2] A. Carrera, J. Solitario, C. Iglesias, Behaviour driven development for multi-agent systems, in: Infrastructure and Tools for Multiagent Systems, 2012, pp. 107–120.

[3] J. Pavon, J. J. Gomez-Sanz, R. Fuentes, The INGENIAS Methodology and Tools, in: B. Henderson-Sellers, P. Giorgini (Eds.), Agent-Oriented Methodologies, IGI Global, 2005.

[4] S. DeLoach, The mase methodology, in: F. Bergenti, M.-P. Gleizes, F. Zambonelli, G. Weiss (Eds.), Methodologies and Software Engineering for Agent Systems, Vol. 11 of Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer US, 2004, pp. 107–125.

[5] L. Padgham, J. Thangarajah, M. Winikoff, Tool support for agent development using the prometheus methodology, in: Quality Software, 2005. (QSIC 2005). Fifth International Conference on, 2005, pp. 383 – 388. doi:10.1109/QSIC.2005.66.

[6] R. Benjamins, Problem-solving methods for diagnosis and their role, International Journal of Expert Systems: Research and Applications 8 (2) (1995) 93–120.

[7] Y. Xiang, F. Jensen, X. Chen, Inference in multiply sectioned bayesian networks: methods and performance comparison, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (3) (2005) 546 –558.

[8] D. Zhang, Multi-agent based control of large-scale complex systems employing distributed dynamic inference engine, Ph.D. thesis, Georgia Institute of Technology (2010).

[9] M. J. Wooldridge, Reasoning about rational agents, Intelligent robotics and autonomous agents, MIT Press, 2000.

[10] P. Kraaijeveld, M. Druzdzel, Genierate: An interactive generator of diagnostic bayesian network models, in: 16th International Workshop on Principles of Diagnosis, 2005, pp. 175–180.

[11] J. Fitzgerald, A. Dennis, Business Data Communications and Networking, John Wiley and Sons, 2008.

[12] F. L. Bellifemine, G. Caire, D. Greenwood, Developing Multi-Agent Systems with JADE, Vol. 5 of Wiley Series in Agent Technology, Wiley, 2007.

[13] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, Mason: A multiagent simulation environment, Simulation 81 (7) (2005) 517–527.

[14] J. Sammon, J.W., A nonlinear mapping for data structure analysis, IEEE Transactions on Computers C-18 (5) (1969) 401 – 409.

[15] L. Paolino, H. Paggi, F. Alonso, G. Lopez, Solving incidents in telecommunications using a multi-agent system, in: Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on, 2011, pp. 303 –308.

[16] M. Sanz-Bobi, M. Castro, J. Santos, Idsai: A distributed system for intrusion detection based on intelligent agents, in: Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on, 2010, pp. 1 –6.

[17] D. P. Cox, Y. Al-Nashif, S. Hariri, Application of autonomic agents for global information grid management and security, Summer Computer Simulation Conference (2007) 1147–1154.

[18] A. Martín, C. León, J. Luque, I. Monedero, A framework for development of integrated intelligent knowledge for management of telecommunication networks, Expert Systems with Applications 39 (10) (2012) 9264 – 9274.

[19] P. Leitão, J. Barbosa, D. Trentesaux, Bio-inspired multi-agent systems for reconfigurable manufacturing systems, Engineering Applications of Artificial Intelligence 25 (5) (2012) 934 – 944.

[20] B. Mendoza, P. Xu, L. Song, A multi-agent model for fault diagnosis in petrochemical plants, in: Sensors Applications Symposium (SAS), 2011 IEEE, 2011, pp. 203 –208.

[21] H. Luo, S. lin Yang, X. jian Hu, X. xuan Hu, Agent oriented intelligent fault diagnosis system using evidence theory, Expert Systems with Applications 39 (3) (2012) 2524 – 2531.

[22] A. Dempster, Upper and lower probabilities induced by a multi-valued mapping, The Annals of Statistics 28 (1967) 325–339.

[23] G. Shafer, A mathematical theory of evidence, Princeton University Press., 1967.