Journal of Network and Computer Applications 37 (2014) 146–154

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

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] Department 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

ARTICLE INFO

Article history:
Received 15 May 2012
Received in revised form
27 September 2012
Accepted 19 November 2012
Available online 2 December 2012

Keywords:
Network management
Agent
Bayesian network
Diagnosis
Uncertainty
BDI

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 paper 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 paper. 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.

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. Highly 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 paper 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 paper is structured as follows. First, Section 2 provides an overview of the MAS design following the Prometheus methodology (Padgham and Winikoff, 2003). 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.

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,

* Corresponding author. Tel.: +34 91 549 57 00x3061.
E-mail addresses: a.carrera@dit.upm.es (Á. Carrera), cif@dit.upm.es (C.A. Iglesias), algarra@tid.es (J. García-Algarra), dusan.kolarik@o2.com (D. Kolařík).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2012.11.004

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 Carrera et al. (2012). This section describes the design and analysis of the MAS. For this purpose, we use the well-known methodology Promotheus (Padgham and Winikoff, 2003), which is compatible with BEAST, although other Agent Oriented Software Engineering (AOSE) methodologies such INGENIAS (Pavon et al., 2005) or MASE (DeLoach, 2004) 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) (Padgham et al., 2005).

Figure 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 (Benjamins, 1995), 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 obtain 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

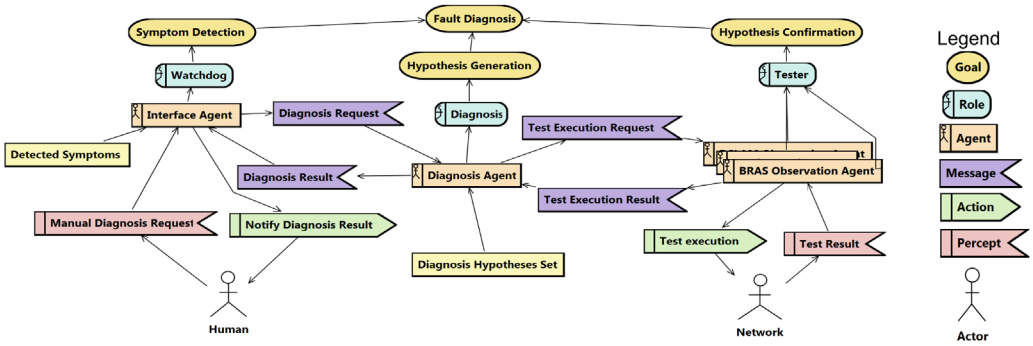

Fig. 1. Analysis overview diagram for a fault diagnosis scenario.