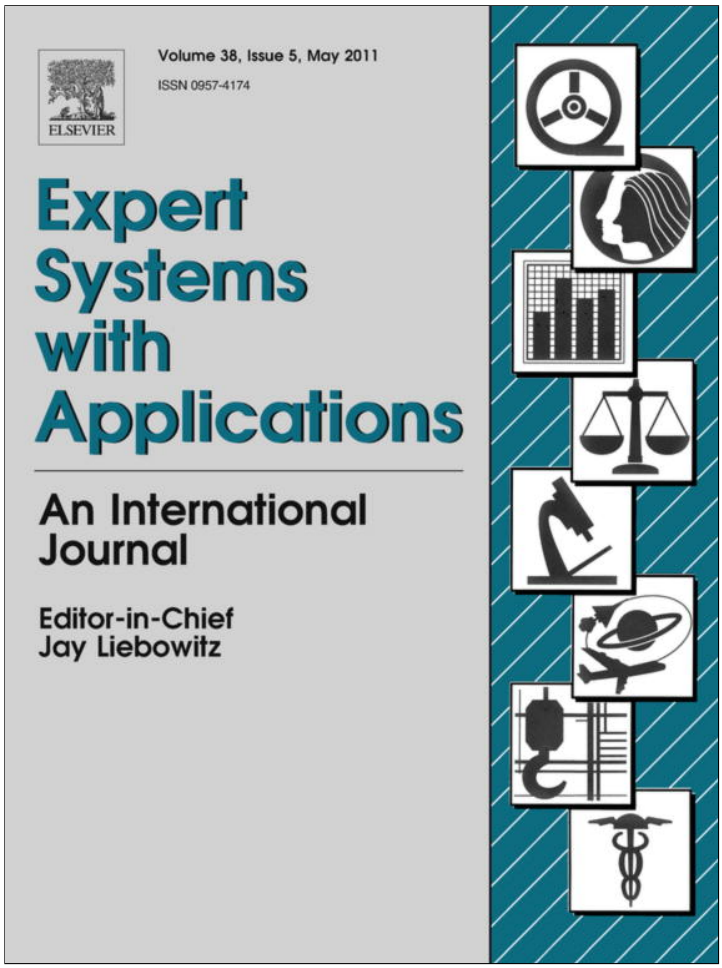

Expert Systems with Applications 38 (2011) 5486–5499

Contents lists available at ScienceDirect

Expert Systems with Applications

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

S-MAS: An adaptive hierarchical distributed multi-agent architecture for blocking malicious SOAP messages within Web Services environments

Cristian I. Pinzón [a], Javier Bajo [b,*], Juan F. De Paz [b], Juan M. Corchado [b]

[a] Universidad Tecnológica de Panamá, Campus Metropolitano "Dr. Víctor Levi Sasso", Vía Ricardo J. Alfaro, Panama
[b] Departamento Informática y Automática Universidad de Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain

ARTICLE INFO

Keywords:
Multi-agent system
Case-based reasoning
Web Services
XML message
Security problems

ABSTRACT

During the last years the use of Web Service-based applications has notably increased. However, the security has not evolved proportionally, which makes these applications vulnerable and objective of attacks. One of the most common attacks requiring novel solutions is the denial of service attack (DoS), caused for the modifications introduced in the XML of the SOAP messages. The specifications of existing security standards do not focus on this type of attack. This article presents the S-MAS architecture as a novel adaptive approach for dealing with DoS attacks in Web Service environments, which represents an alternative to the existing centralized solutions. S-MAS proposes a distributed hierarchical multi-agent architecture that implements a classification mechanism in two phases. The main benefits of the approach are the distributed capabilities of the multi-agent systems and the self-adaption ability to the changes that occur in the patterns of attack. A prototype of the architecture was developed and the results obtained are presented in this study.

1. Introduction

Web Services have become increasingly relevant not only within private networks for companies and organizations, but also at the level of inter-communication. This trend to inter-communication in the Web Services has made the security a key element in open architectures. However, basic Web Service specifications themselves do not address any security topics. Several additional specifications as WS-Security (Oasis, 2004), WS-SecurityPolicy (Della-Libera & Zolfonoon, 2005), WS-Trust (Anderson et al., 2005), WS-SecureConversation (Anderson et al., 2004) etc. for Web Services security exists, but all these standards focus on the aspects of message integrity and confidentiality and user authentication and authorization. Then, it is necessary to investigate in novel method to protect the servers from denial of services attacks (DoS), which cause malicious or altered Web Services, and affect the availability of the Web Services (Gruschka & Luttenberger, 2006). DoS attacks are due to the fact that XML messages must be parsed in the server, which opens the possibility of an attack if the messages themselves are not well structured or if they include some type of malicious code. Resources available in the server (memory and CPU cycles) of the provider can be drastically reduced or exhausted while a malicious SOAP message is being parsed. A DoS attack is successfully carried out when it manages to severely compromise legitimate user access to services and resources.

Some approaches focus on preventing DoS attacks to Web Services architectures (Bebawy, Sabry, El-Kassas, Hanna, & Youssef, 2005; Chonka, Zhou, & Xiang, 2009; Gruschka & Luttenberger, 2006; Im & Song, 2005; Loh, Yau, Wong, & Ho, 2006; Padmanabhuni, Singh, Kumar, & Chatterjee, 2006; Srivatsa, Iyengar, Yin, & Liu, 2008; Wang, 2006; Ye, 2008; Yee, Shin, & Rao, 2007), but present as main disadvantage their low capacity to adapt themselves to the changes in the patterns of attack, which causes a reduction of the effectiveness of these methods when slight variations in the behaviours of the known attacks happen or when new attacks appear. Moreover, most of the existing approaches are based on a centralized perspective, which provides facilities for DoS attacks. In this sense, and focusing on performance aspects, centralized approaches can become a bottleneck when security is vulnered, causing a reduction of the overall performance of the application. Another approaches focus on providing solutions to DoS attacks in Web Services environments with a perspective similar to traditional layer 2–4 firewalls and application level firewalls which no longer viewed as an effective way for providing a solution to the Web Services. The use of Web Services over HTTP makes it hard to use traditional layer 2–4 firewalls to block malicious web services traffic (Bebawy et al., 2005).

Taking into account the limitations of the existing approaches and the particularities of the new trends in DoS attacks, this paper presents a distributed hierarchical multi-agent architecture for

* Corresponding author. Tel.: +34 639771985.
E-mail addresses: cristian_ivanp@usal.es (C.I. Pinzón), jbajope@usal.es, jbajope@upsa.es (J. Bajo), fcofds@usal.es (J.F. De Paz), corchado@usal.es (J.M. Corchado).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.10.088

dealing with DoS attacks in Web Service environments. The proposed architecture is based in our previous research in SQL injection attacks (Bajo, Corchado, Pinzón, Paz, & Pérez-Lancho, 2008; PinzóN, Paz, & Bajo, 2008) where a multi-agent architecture SQL-MAS was developed. In this way, some resources are reused and the knowledge acquired in this previous work is adapted to get an evolution of the architecture. The S-MAS architecture has a four-tiered hierarchical design that is better capable of task distribution and error recovery. The classification mechanism integrated within the multi-agent architecture has also evolved, incorporating a two-phase strategy to classify SOAP messages. The first phase applies the initial filter for detecting simple attacks without requiring an excessive amount of resources. The second phase involves a more complex process which ends up using a significantly higher amount of resources. In this way, a strategy in two-phase improves the overall response time of the classification mechanism, facilitating a quick classification of those incoming SOAP messages with significative features during the first phase. The second phase is executed only for those SOAP messages with complex characteristics identified as suspicious during the first phase and requiring a more detailed evaluation. Each of the phases incorporates a CBR–BDI (Laza, Pavó, & Corchado, 2003) agent with reasoning, learning and adaptation capabilities.

The idea of a CBR mechanism is to exploit the experience gained from similar problems in the past and then adapt successful solutions to the current problem. The CBR engine initiates what is known as the CBR cycle, which is comprised of four stages. The approach presented in this paper proposes a classifier agent for the first phase (CBRMAS-L1) that incorporates a classification strategy based on a classification tree, and a classifier agent for the second phase (CBRMAS-L2) that incorporates a neural network. Each of these classification strategies is incorporated into the respective re-use stage of the CBR cycle integrated into the corresponding agent. As a result, the system can learn and adapt to the attacks and the changes in the techniques used in the attacks. The model proposed in this study is innovative, since proposes a new perspective to address the DoS attacks problem in Web Services environments. The model is not aimed at replacing the existing security solutions, but to be established as an additional layer to cover the lacks of the previous approaches.

The rest of the paper is structured as follows: Section 2 presents the problem that has prompted most of this research work. Section 3 focuses on the details of the multi-agent architecture; Section 4 explains in detail the classification model designed in two phases. Section 5 describes a case study based on a multi-agent system that provides Web Services in a shopping mall. Finally, the final results and conclusion are presented in Section 6.

2. Work web service security problem description

One of the most frequent techniques of a DoS attack is to exhaust available resources (memory, CPU cycles, and bandwidth) on the host server. The probability of a DoS attack increases with applications providing Web Services because of their intrinsic use of the XML standard. The server uses a parser, such as DOM and Xerces to syntactically analyze all incoming XML formatted SOAP messages. When the server draws too much of its available resources to parse SOAP messages that are either poorly written or include a malicious code, it risks becoming completely blocked.

Attacks usually occur when the SOAP message either comes from a malicious user or is intercepted during its transmission by a malicious node that introduces different kinds of attacks.

The following list contains descriptions of some known types of attacks that can result in a DoS attack, as noted in Jensen, Gruschka, Herkenhoner, and Luttenberger (2007), Loh et al. (2006), Yee et al. (2007) .

- *Oversize payload:* When executed, it reduces or eliminates the availability of a web service while the CPU, memory or bandwidth are being tied up by a massive mailing with a large payload.
- *Coercive parsing:* Just like a message written with XML, an XML parser can analyze a complex format and lead to an attack in which a service is rejected because the memory and processing resources are being used up.
- *Injection XML:* This is based on the ability to modify the structure of an XML document when an unfiltered user entry goes directly to the XML stream and the message is captured and modified during its transmission.
- *SOAP header attack:* Some SOAP message headers are overwritten while they are passing through different nodes before arriving at their destination. It is possible to modify certain fields with malicious code.
- *Replay attack:* Sent messages are completely valid, but they are sent en masse over short periods of time in order to overload the web service.

All web service security standards focus on strategies independent from DoS attacks (Gruschka & Luttenberger, 2006). In the following, we will revise those works that focus on denial of web service attacks and will compare to our approach as shown in Table 1.

Im and Song present and adaptive approach (Im & Song, 2005), which extends the proposal of Schuba et al. (1997), that is based on a tool located in a firewall aimed at monitoring and detecting SYN flooding attacks. This tool examines the TCP packages and categorizes the IP addresses into different states. The extension proposed by Im and Song (2005) adapts the initial approach to Project the Web Services and introduces some improvements consisting on adding priorities to the states and examining the incoming and outgoing packages of the server. Only SYN flooding attacks in web services environments are considered. Bebawy et al. propose the tool "Nedgty" (Bebawy et al., 2005) based on a Web Service firewall model. The target operating system for Nedgty is the Linux OS. Nedgty secures Web Services by applying business specific rules in a centralized manner. It works at the application level as a stand-alone application and its design is a hybrid of a fully fledged proxy. This solution secures Web Services by intercepting packets going to the server, determining the Web Services specific packets, and checking them for any malicious content. In addition, it filters out unauthorized requests that originate from IP addresses that are not allowed. Wang (2006) introduce the session based ideas into the access of Web Services and develop a fair share based filtering algorithm. The proposed architecture is based on the two-phase access model. In the first phase, the Web service container will bind the related user information (user name) to a customized WSDL version for the requester user and return this file. In the second phase, when the user sends a Web service request, the Web Services container can recognize the user information. Based on the current working load information, a filter component will decide that whether or not this request can be passed into the Web services engine. The algorithm and a set of parameters are defined. A XML Firewall is proposed by Loh et al. (2006). The architecture of the XML Firewall is divided into three modules, namely Core Engine, Administrative Interface, and Database. The Core engine is the main component that processes and handles SOAP messages. Messages that are sent to a Web Service are intercepted and parsed to check the validity and the authenticity of the contents. If the contents of the messages do not conform to the policies that have been set, the messages will be dropped by

Table 1
Comparison of selected models approaches vs. S-MAS.

	DoS defense mechanism (Im & Song, 2005)	Nedgty (Bebawy et al., 2005)	A fair share based filtering algorithm (Wang, 2006)	XML Firewall (Loh et al., 2006)	CheckWay Gateway (Gruschka & Luttenberger, 2006)	PreSODoS (Padmanabhuni et al., 2006)	ID/IP framework (Yee et al., 2007)	Twofold mechanism (Srivatsa et al., 2008)	DDoS and XDoS defense system (Ye, 2008)	SOTA & XDetector (Chonka et al., 2009)	S-MAS
Distributed approach	No	No	No	No	No	No	No	No	No	Yes	Yes
Learning ability	No	No	No	No	No	No	Yes	No	No	Yes	Yes
Adaptive ability	No	No	No	No	No	No	Yes	No	No	No	Yes
Balances the workload	No	No	No	No	No	No	–	–	No	Yes	Yes
Tolerance to failure	No	–	No	No	Yes	–	–	Yes	–	–	Yes
Scalability	–	Yes	Yes	Yes	–	Yes	Yes	–	Yes	Yes	Yes
Positive and negative false	–	–	–	No	–	–	–	–	–	Yes	Yes
Time response	–	–	Nearly real time	–	Nearly real time	–	–	Nearly real time	Nearly real time	–	Nearly real time
Ubiquity	No	No	No		No	No	No	No	No	No	Yes

the firewall. Three successfully implemented filtering policies, namely message size filtering, syntax parsing, and XML schema validation have been tested with valid and invalid SOAP messages. Gruschka and Luttenberger (2006) propose an application level gateway system "Checkway". They focus on a full grammatical validation of messages by Checkway before forwarding them to the server. To do this, they consider that Web Service messages are XML documents and these are usually defined by an XML Schema, written in the XML Schema definition language—a grammar language for XML. Checkway generates an XML Schema from a Web Service description and validates all Web Service messages against this schema. The approach presents a centralized model oriented to detect concrete type of attack inside Web Services. Padmanabhuni et al. (2006) propose PreSODoS. This framework relies on content introspection to detect any XDoS possibility. PreSODoS use a Patricia Trie based representation so that the schemas and the request messages can be compared and validated in a performance efficient manner. PreSODoS leverages existing security infrastructures. An adaptive framework for the prevention and detection of intrusions was presented in Yee et al. (2007). Based on a hybrid focus that combines agents, data mining and diffused logic, it is supposed to filter attacks that are either already known or new. Agents that act as sensors are used to detect violations to the normal profile using the data mining technique such as clustering, association rules and sequential association rules. The anomalies are then further analyzed using fuzzy logic to determine genuine attacks so as to reduce false alarms. If an attack is being detected, a specific component will act to prevent the attack from happening. An approach to handle DoS attacks by using a twofold mechanism is presented by Srivatsa et al. (2008). First, an admission control is performed to limit the number of concurrent clients served by the online service. Admission control is based on port hiding that renders the online service invisible to unauthorized clients by hiding the port number on which the service accepts incoming requests. Second, a congestion control is performed on admitted clients to allocate more resources to good clients. Congestion control is achieved by adaptively setting a client's priority level in response to the client's requests in a way that can incorporate application-level semantics. The experiments show that the techniques incur low performance overhead. In addition, the proposed techniques can be easily deployed into existing Web/application servers. A approach to countering DDoS and XDoS Attacks against Web Services is presented by Ye (2008). The system carries out request message authentication and validation before the requests are processed by the Web Services providers. The scheme has two modes: the normal mode and the under-attack mode. A component called "operations provider" decides which mode the system works in. In the under-attack mode, the service requests need to be authenticated and validated before being processed. Since the system is constructed from Web Services, it can be formed and reconfigured easily. Finally, a recent solution proposed by Chonka et al. (2009) presents a Service Oriented Traceback Architecture (SOTA) to cooperate with a filter defense system, called XDetector. XDetector is a Back Propagation Neural Network, trained to detect and filter XDoS attack message. SOTA is a traceback system that is constructed on the basis of Web Services and is able to traceback to the source of the malicious message. Once an attack has been discovered and the attacker's identity known, XDetector can filter out these attack messages.

Table 1 presents a comparison of S-MAS with the current approaches aimed at detecting DoS attacks in Web Services environments. Those parameters that could not be evaluated are marked with a hyphen.

According to the results shown in Table 2, S-MAS outperforms the existing models with respect to:

- *Distributed approach:* S-MAS is based on a multi-agent architecture that can execute tasks derived from the classification process in a distributed way.
- *Adaptive ability:* S-MAS includes two types of intelligent CBR-MAS agents that were designed to learn and adapt to changes in attack patterns, new attacks, and types of user behaviour.

Table 2
Problem description first phase – CBRSOAP-L1.

Fields	Type	
IDService	Int	*i*
Subnet mask	String	*m*
SizeMessage	Int	*s*
NTimeRouting	Int	*n*
LengthSOAPAction	Int	*l*
TFMessageSent	Int	*w*

- *Balances the workload:* S-MAS was designed to distribute the classification task load throughout the various layers of the hierarchical architecture.
- *Tolerance to failure:* S-MAS has a hierarchical design that can facilitate error recovery through the instantiation of new agents.
- *Scalability*: S-MAS is capable of growing (by means of the instantiation of new agents) according to the needs of its environment.
- *Ubiquity*: S-MAS provides a ubiquitous alert mechanism to notify security personnel in the event of an attack.

Some aspects of S-MAS, such as response time or the required initial learning period can be considered as a disadvantage when compared to other solutions. Nevertheless, S-MAS provides a much more efficient classification once the system acquires experience, and a reasonably low response time. The S-MAS architecture presents novel characteristics that have not heretofore been considered in previous approaches. The next section presents the S-MAS architecture in greater detail.

3. S-MAS architecture

Agents are characterized by their autonomy; which gives them the ability to work independently in real-time environments (Carrascosa, Bajo, Julian, Corchado, & Botti, 2008). Furthermore, when they are integrated within a multi-agent system they can also offer collaborative and distributed assistance in carrying out tasks (Corchado, Bajo, & Abraham, 2008). One of the main characteristics of the multi-agent architecture proposed in this paper is the incorporation of CBR–BDI (Laza et al., 2003) deliberative agents, CBRMAS classifier agents, that use an intrusion detection technique known as anomaly detection. In order to carry out this type of detection, it is necessary to extract information from the structure and content of the SOAP messages, the message processing tasks, and any activity among web service users. Our proposal is a distributed, hierarchical multi-agent architecture integrated for four levels with distinct BDI agents called "S-MAS". The hierarchical structure makes it possible to distribute tasks on the different levels of the architectures and to define specific responsibilities. The architecture S-MAS presented in Fig. 1 shows the four levels with BDI agents organized according to their roles.

- *Traffic agents:* Capture any traffic directed towards the server. JPCAP (Fujii, 2000) is the API used to identify and capture any traffic that contains SOAP message packets. The captured SOAP messages are sent to the next layer in order to carry out the classification process. In addition to these tasks, Traffic agents use an IP register to monitor user activities. This type of monitoring makes it possible to identify suspicious activities similar to message replication attacks.

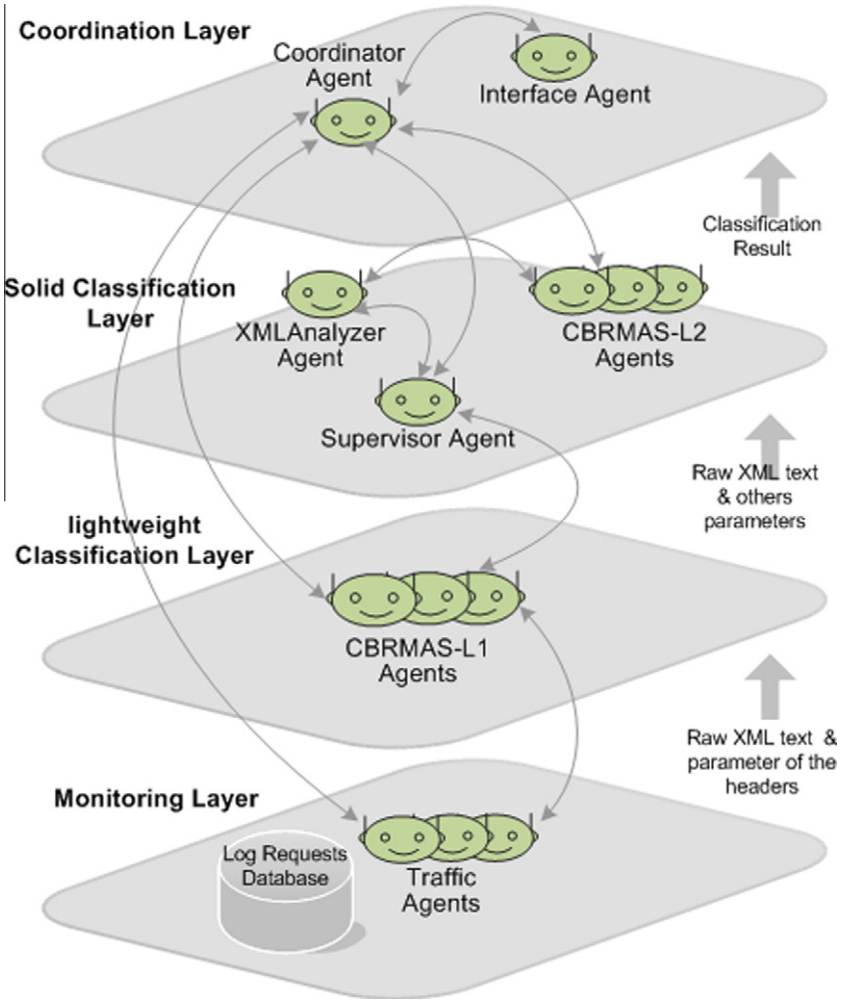

Fig. 1. Design of the multi-agent architecture proposed.

- *CBRMAS-L1 agents:* These advanced agents are the first-part of the core of the multi-agent architecture. These CBR–BDI agents are located on layer 2 of the architecture and are in charge of executing the first phase of the classification process based on the data sent by the Traffic agents. These agents initiate a classification by incorporating a CBR engine that in turn incorporates a decision tree strategy in the re-use phase. The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources.
- *CBRMAS-L2 agents:* These CBR–BDI agents complete the classification process from layer 3 of the architecture. These advanced agents are the second-part of the core of the multi-agent architecture. In order to initiate this phase, it is necessary to have previously started a syntactic analysis on the SOAP message to extract the required data. This syntactic analysis is performed by the XMLAnalyzer Agent. Once the data have been extracted from the message, a CBR mechanism is initiated by using a Multilayer Perceptron (MLP) neural network in the re-use phase.
- *Supervisor agent:* This agent supervises the XMLAnalyzer agent since there still exists the possibility of an attack during the syntactic processing of the SOAP message. This agent is located in layer 3 of the architecture.
- *XMLAnalyzer agent:* This agent executes the syntactic analysis of the SOAP message. The analysis is performed using SAX (Brownell, 2002) as parser. Because SAX is an event driven API, it is most efficient primarily with regards to memory usage, and strong enough to deal with attack techniques. The data extracted from the syntactic analysis are sent to the CBRMAS-L2 agents. This agent is also located on layer 3 of the architecture.
- *Coordinator agent:* This agent is in charge of supervising the correct overall functioning of the architecture. Additionally, it oversees the classification process. Each time a classification is tagged as suspicious, the agent interacts with the Interface Agent to request an expert review. Finally, this agent controls the alert mechanism and coordinates the actions required for responding to this type of attack. This agent is located on layer 4 of the architecture.
- *Interface agent:* This agent was designed to function in different devices (PDA, Laptop, and Workstation). It facilitates ubiquitous communication with the security personnel when an alert has been sent by the Coordinator Agent. Additionally, it facilitates the process of adjusting the global configuration of the architecture. This agent is also located in the highest layer of the architecture.

The following section describes the functionality of the classifier agents located layers 2 and 3 of the proposed hierarchical structure.

4. Mechanism for the classification of SOAP message attack

The application of agents and multi-agent systems facilitates taking advantage of the inherent capabilities of the agents. Nevertheless, it is possible to increase the reasoning and learning capabilities by incorporating a case-based reasoning (CBR) mechanism into the agents. In the case at hand, a CBR classifier agent (CBR–BDI) is responsible for classifying the incoming SOAP messages. In the BDI (Beliefs Desires Intentions) model, the internal structure of an agent and its capacity to choose is based on mental aptitudes: agent behaviour is composed of beliefs, desires, and intentions (Laza et al., 2003). A BDI architecture has the advantage of being intuitive and capable of rather simply identifying the process of decision-making and how to perform it.

Case-based reasoning is a type of reasoning based on the use of past experiences (Aamodt & Plaza, 1994). The purpose of case-based reasoning systems is to solve new problems by adapting solutions that have been used to solve similar problems in the past. The fundamental concept when working with case-based reasoning is the concept of case. A case can be defined as a past experience, and is composed of three elements: A problem description which describes the initial problem, a solution which provides the sequence of actions carried out in order to solve the problem, and the final state which describes the state achieved once the solution was applied. The way in which cases are managed is known as the case-based reasoning cycle. This CBR cycle consists of four sequential steps: retrieve, reuse, revise and retain (Aamodt & Plaza, 1994). A CBR engine requires the use of a database with which it can generate models such as the solution of a new problem based on past experience.

In the specific case of SOAP messages, a case memory is managed for each service offered by the Web Service environment, which permits it to handle each incoming message based on the particular characteristics of each web service available. Each new SOAP message sent to the architecture is classified as a new case study object. Focusing on the problem that is of interest to us, we will represent a typical SOAP message which consists of a type of wrapping that contains an optional heading and a mandatory body of text with a useful message load, as depicted in Fig. 2a and b.

Based on the structure and content of the SOAP message, the message processing tasks, and any activity among web service users we can obtain a series of descriptive fields.

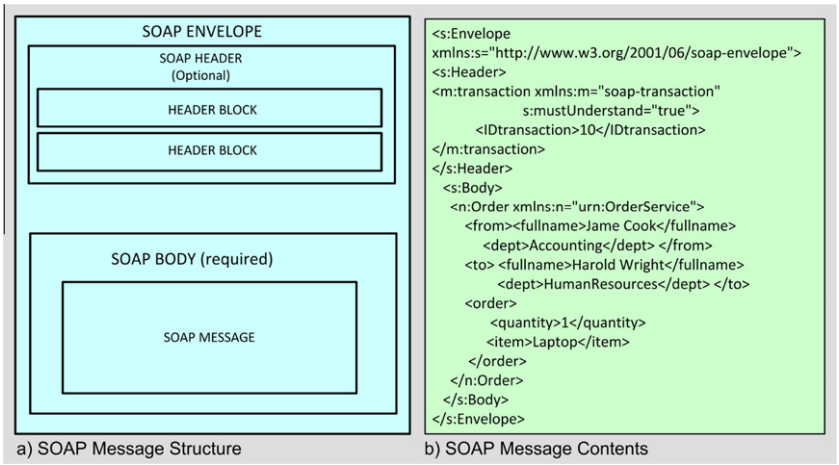

Fig. 2. (a) SOAP message structure and (b) SOAP message content.

4.1. First phase of the mechanism of classification – CBRMAS-L1 agents

The main goal of this initial phase is to carry out an effective classification, but without requiring an excessive amount of resources and time. As a CBR strategy is used, it is necessary to define the case structure used by the CBRMAS-L1 agents. The fields of the case are obtained from the headers of the packages of the HTTP/TCP-IP transport protocol. Table 2 shows the fields taken into consideration to describe the problem.

As can be seen in Table 2, the description of a case is given by the tuple $c = (i, m, s, n, l, w, R/c_{.im}, x^p, x^r)$, where i represents the service identifier, m the subnet mask, s the message length, n the number of seconds for the travel of the message, l the length of the header SoapAction, w the elapsed time from the arrival of the last n messages, $R/c_{.im}$ is the solution provided by the decision tree associated to the service and to the subnet mask, x^p represents the class predicted by the CBR strategy $x^p \in X = \{a, g, u\}$, where a, g, u represent the values attack, good and undefined, x^r is the real class $x^r \in X = \{a, g, u\}$.

The CBR strategy is integrated into a BDI agent, obtaining a CBR–BDI agent. The integration of the CBR system and the BDI agent is defined as follows: believes – problem description and rules; intentions – set of relieves and rules that represent the state transitions required to achieve the final state; desires – $X = \{a, g, u\}$. The initial state is defined by means of the set of believes that store the values for the subnet mask and the service web identifier, $(i, m, \phi, \phi, \phi, \phi)$. The intermediate states describe the decision process executed, taking into account the application of rules over the set of rules.

The cases memory contains a set of cases $C=\{c\}$ and is fragmented for each of the Web Services available in the server. This structure facilitates the depuration and analysis of the services in an independent manner. Separately to the cases memory, the agent incorporates a rules memory, constructed as a set of inductive rules defined as $R = \{r_1, \ldots, r_l\}$ with $r_i = (l_1 \wedge \cdots \wedge l_m) \rightarrow x_j$ where $l_s = (d_{ts}, o_s, \Re)/d_{ts} \in \{i, m, s, n, l, w, x^p, x^r\}, o_s \in O$, with $O = \{=, \neq, >, <, \leqslant, \geqslant\}$, $x_j \in X$. The rules memory is also fragmented for each of the services and for each of the subnet mask, in a way that $R/C_{.im}$ represents the rules associated to those cases belonging to the service i and the subnet mask m. For notation considerations, to identify a property of a case, we use the case, a point and the property. For example, $c_{j.m}$ represents the property m (subnet mask) of the case j.

When the agent receives a request to classify a new case c_{n+1}, a new execution of a CBR cycle is carried out. The following paragraphs describe the stages of a CBR cycle executed by a CBRMAS-L1 agent in charge of a first phase classification.

- *Retrieve*: During this stage, those cases associated to the requested web service and the corresponding rules memory are retrieved. The storage and recovery of rules from the rules memory facilitates a notably reduction of the process time for the classification. The retrieve strategy is carried out as follows:
  - If there is not tree associated to the service and the subnet mask, then it is necessary to recover the cases for the service and the subnet mask:

$$c_{.im} = f_s(C) = \{c_{j.im} \in C / c_{j.i} = c_{n+1.i}, c_{j.m} = c_{n+1.m}\} \quad (1)$$

    where $C_{j.i}$ represents the case j and i the service identifier.
  - The rules memory associated with the set of cases $R/c_{.im}$ is retrieved.
- *Reuse*: Knowledge extraction is especially important when complex algorithms that use hard computing techniques and that generate models in an automatic way are used. Human experts are much confident when they know exactly why or at least how a solution to a problem has been calculated. CART is a non-parametric statistical method for extraction of knowledge in classifications. The extracted information is represented in a binary decision tree, which allows individuals to be classified from the root node. Keeping the kind of dependent variable in mind, CART can be separated into two types: classification tree, if the dependent variable is categorical; and regression tree in the case of a continuous dependent variable.
  The reuse stage is only executed if not decision tree $R/c_{.im}$ associated to the cases $c_{.im}$ is available, and in order to do so, the rules are generated using the CART algorithm. $R/c_{.im} = CART(c_{.im})$ where $R/c_{.im}$ is the rules memory associated to the service identifier and to the subnet mask. The CART algorithm has been modified in order to have an automatic discretization of the values to a set of categories. The modification includes a first step to normalize the variable into the interval [0,1] and then, the values are discretized into one of the following categories depending on the closest value {very low = 0.1, low = 0.3, medium = 0.5, high = 0.7 and very high = 0.9}. This way, the generation of rules using the CART algorithm is more efficient than working with a greater level of categories. The discretization is only carried out for the variables s, n, l, w.
- *Revise*: Once the set of rules has been retriever, the classification for the case $c_{n+1.im}$ is obtained using the set of rules that previously classified the elements of the same type $c_{n+1.x^p} = R/c_{.im}(c_{n+1})$. If $r_i \in R/c_{.im}$ then, it is the rule that classifies c_{n+1}. The new case is classified as follows:
  - If $m_i > \mu_1 \| \#\{c_j \in C_{r_j}/c_{j.x^p} = u\} > \mu_2$ then, it is necessary to execute the CBR of the second phase. Where $C_{r_i} \subseteq C$ is the set of cases classified for r_i and m_i represents the percentage of misclassified cases of C_{r_j} using the rule r_i. # represents the number of elements of the set. The general idea is to verify if the error rate of the rule exceeds a certain threshold, and then, verify that the number of cases belonging to the set of elements classified using the rule not exceeds a certain threshold defined as a function of the total number of elements in C_{r_j}
  - Else if $\#\{c_j \in C_{r_j}/c_{j.x^p} = g\}/\#C_{r_j} > \alpha_g$ then the case is classified as good and the revision finishes.
  - Else if $\#\{c_j \in C_{r_j}/c_{j.x^p} = u\}/\#C_{r_j} > \alpha_s$ the case is classified as suspicious and the second phase classification mechanism is executed.
  - Else if $\#\{c_j \in C_{r_j}/c_{j.x^p} = a\}/\#C_{r_j} > \alpha_a$ the case is classified as attack and the revision finishes.
- *Retain*: If the set of rules was generated because it did not previously exist, then $R/c_{.im}$ is stored in the rules memory if the classification obtained was good. If the classification was erroneous and the misclassification was detected by an expert or if the CBRMAS-L2 of the second phase was invoked, then it is necessary to regenerate the decision tree: $R/c_{.im} = CART(c_{.im} \cup c_{n+1})$.

Fig. 3 shows the stages of the CBR cycle for the CBRMAS-L1 agent.

4.2. Second phase of the mechanism of classification – CBRMAS-L2 agents

The second phase of the mechanism of classification is carried out by means of CBRMAS-L2 agents. As these agents are CBR–BDI agents, it is necessary to provide a case description. The fields are extracted from the SOAP message and provide the case description for the CBRMAS-L2 agents. Table 3 presents the fields used in describing the problem for the CBR in this layer. Applying the nomenclature shown in the table above, each case description is given by the following tuple:

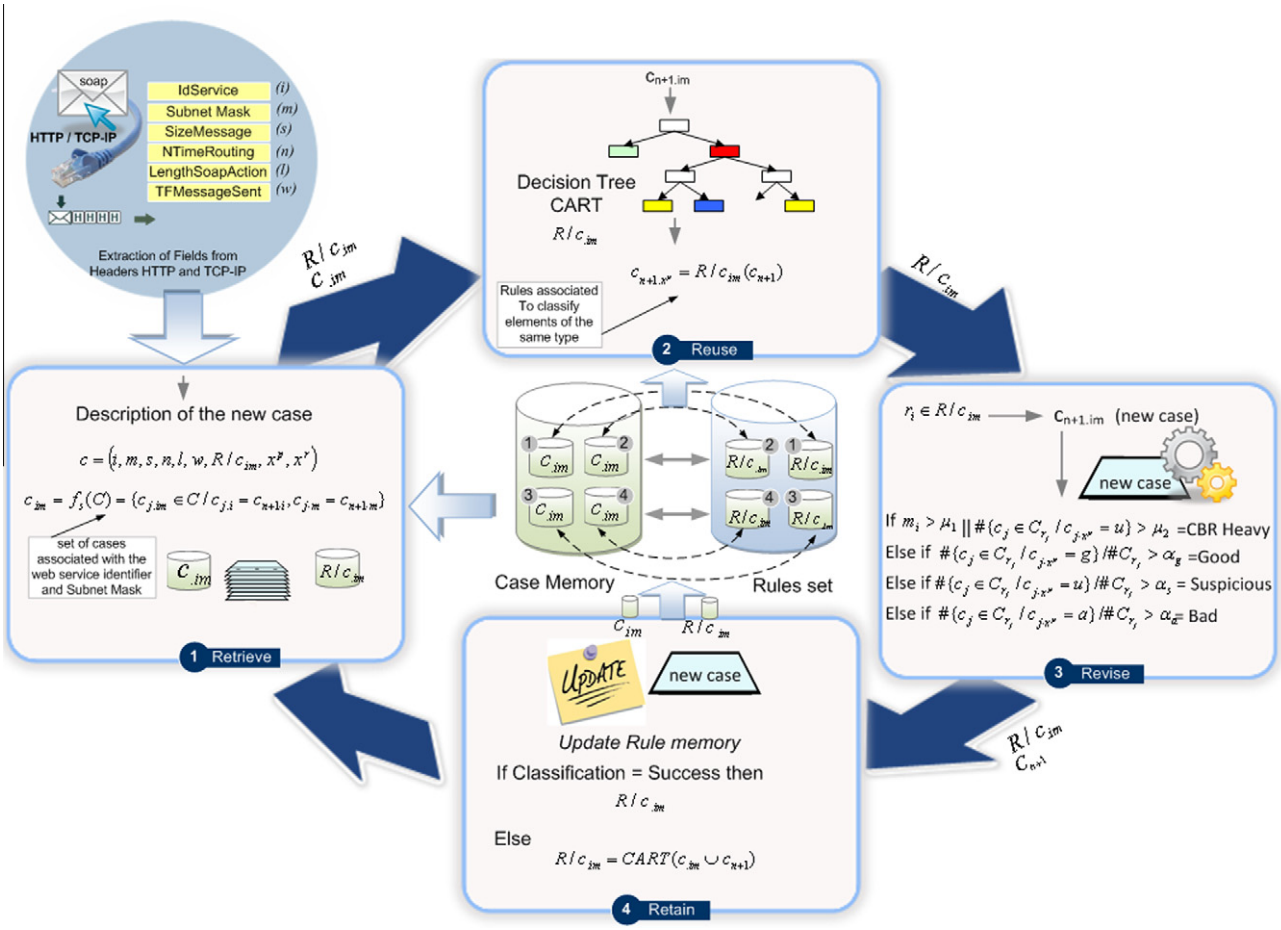

Fig. 3. Stages of the CBR cycle of a CBRMAS-L1 agent in the first phase of the classification mechanism.

Table 3
Case description second phase – CBRSOAP-L1 agents.

Fields	Type	Variable
IDService	Int	i
MaskSubnet	String	m
SizeMessage	Int	s
NTimeRouting	Int	n
LengthSOAPAction	Int	l
MustUnderstandTrue	Boolean	u
NumberHeaderBlock	Int	h
NElementsBody	Int	b
NestingDepthElements	Int	d
NXMLTagRepeated	Int	t
NLeafNodesBody	Int	f
NAttributesDeclared	Int	a
CPUTimeParsing	Int	c
SizeKbMemoryParser	Int	k

$$c = (i, m, s, n, l, u, h, b, d, t, f, a, c, k, P/c_{.im}, x^p, x^r) \quad (1)$$

For each incoming message received by the agent and requiring classification, we will consider both the class that the agent predicts and the class to which the message actually belongs. x^p represents the class predicted by the CBRMAS-L2 agents belonging to the group. $x^p \in X = \{a, g, u\}$; g and u represent attack, good and undefined, respectively; and x^r is the class to which the attack actually belongs $x^r \in X = \{a, g, u\}$, $P/c_{.im}$ is the solution provided by the neural network MLP associated to service i and subnet mask m.

The reasoning memory used by the agent is defined by the following expression: $P = \{p_1, \ldots, p_n\}$ and is implemented by means of a MLP neural network. Each P_i is a reasoning memory related to a group of cases dependent of the service and subnet mask of the client. The Multilayer Perceptron (MLP) is the most widely applied and researched artificial neural network (ANN) model. MLP networks implement mappings from input space to output space and are normally applied to supervised learning tasks (Gallagher & Downs, 2003). The Sigmoidal function was selected as the MLP activation function, with a range of values in the interval [0,1]. It is used to detect if the SOAP message is classified as an attack or not. The value 0 represents a legal message (non attack) and 1 a malicious message (attack). The sigmoidal activation function is given by

$$f(x) = \frac{1}{1 + e^{-ax}} \quad (2)$$

The CBR mechanism executes the following phases:

Retrieve: Retrieves the cases that are most similar to the current problem, considering both the type of Web service to which the message belongs and the subnet mask that contains the message.

- Expression 3 is used to select cases from the case memory based on the type of Web service and the subnet mask.

$$c_{.im} = f_s(C) = \{c_j \in C / c_{j.i} = c_{n+1.i}, c_{j.m} = c_{n+1.m}\} \quad (3)$$

- Once the similar cases have been recovered, the neural network MLP $P/c_{.im}$ associated to service i and subnet mask m is then recovered.

Reuse: The classification of the message is begun in this phase, based on the subnet mask and the recovered cases. It is only

necessary to retrain the neural network when it does not have previous training. The entries for the neural network correspond to the case elements *s*, *n*, *l*, *u*, *h*, *b*, *d*, *t*, *f*, *a*, *c*, *k*. Because the neurons exiting from the hidden layer of the neural network contain sigmoidal neurons with values between [0,1], the incoming variables are redefined so that their range falls between [0.2–0.8]. This transformation is necessary because the network does not deal with values that fall outside of this range. The outgoing values are similarly limited to the range of [0.2, 0.8] with the value 0.2 corresponding to a non-attack and the value 0.8 corresponding to an attack. The training for the network is carried out by the error Backpropagation algorithm (Lecun, Bottou, Orr, & MÜller, 1998). The weights and biases for the neurons at the exit layer are updated by following equations:

$$w_{kj}^{p}(t+1) = w_{kj}^{p}(t) + \eta(d_k^p - y_k^p)(1-y_k^p)y_k^p y_j^p + \mu(w_{kj}^{p}(t) - w_{kj}^{p}(t-1)) \quad (4)$$

$$\theta_k^p(t+1) = \theta_k^p(t) + \eta(d_k^p - y_k^p)(1-y_k^p)y_k^p + \mu(\theta_k^p(t) - \theta_k^p(t-1)) \quad (5)$$

The neurons at the intermediate layer are updated by following a procedure similar to the previous case using the following equations:

$$w_{ji}^{p}(t+1) = w_{ji}^{p}(t) + \eta(1-y_j^p)y_j^p\left(\sum_{k=1}^{M}(d_k^p - y_k^p)(1-y_k^p)y_k^p w_{kj}\right)x_i^p + \mu(w_{ji}^{p}(t) - w_{ji}^{p}(t-1)) \quad (6)$$

$$\theta_j^p(t+1) = \theta_j^p(t) + \eta(1-y_j^p)y_j^p\left(\sum_{k=1}^{M}(d_k^p - y_k^p)(1-y_k^p)y_k^p w_{kj}\right) + \mu(\theta_j^p(t) - \theta_j^p(t-1)) \quad (7)$$

where w_{kj}^{p} represents the weight that joins neuron j from the intermediate layer with neuron k from the exit layer, t the moment of time and p the pattern in question. d_k^p represents the desired value, y_k^p the value obtained for neuron k from the exit layer, y_j^p the value obtained for neuron j from the intermediate layer, η the learning rate and μ the momentum. θ_k^p represents the bia value k from the exit layer. The variables for the intermediate layer are defined analogously, keeping in mind that i represents the neuron from the entrance level, j is the neuron from the intermediate level, M is the number of neurons from the exit layer.

When a previously trained network is already available, the message classification process is carried out in the revise phase. If a previously trained network is not available, the training is carried out following the entire procedure beginning with the cases related to the service and subnet mask, as shown in Eq. (8).

$$p_r = MLP^t(c_{.im}) \quad (8)$$

Revise: This phase reviews the classification performed in the previous phase. The value obtained by exiting the network $y = P_r^e(c_{n+1})$ may yield the following situations:

- If $y > \mu_1$ then it is considered an attack.
- Otherwise, if $y < \mu_2$, then the message is considered a non-attack or legal.
- Otherwise, the message is marked as suspicious and is filtered for subsequent revision by a human expert. To facilitate the revision, an analysis of the neural network sensibility is shown so that the relevance of the entrances can be determined with respect to the predicted value.

Retain: If the result of the classification is suspicious or if the administrator identifies the classification as erroneous, then the network $P/c_{.im}$ repeats the training by incorporating a new case and following the BackPropagation training algorithm.

$$p_r = MLP^t(c_{.im} \cup c_{n+1}) \quad (9)$$

Fig. 4 shows the stages of the CBR cycle for the CBRMAS-L2 Agents, which constitute the second and last phase of the classification mechanism. The next section describes a case study developed to evaluate a prototype of the architecture presented in this paper.

5. Case study

A case study was proposed to test the effectiveness of a S-MAS prototype. The prototype was evaluated by a previously developed multi-agent system installed in the Tormes shopping mall (Corchado, Bajo, DePaz, & Rodríguez, 2009). An intelligent environment based on the use of Wi-Fi, Bluetooth and RFID and handheld devices was implemented in this mall. The intelligent environment improves the services offered in the shopping mall by providing personalized services through handheld devices. The clients can receive personalized promotions, recommendations about products or shops and guiding suggestions. They can also receive news or advises of their particular interest, or information about other clients with similar preferences (with whom they can communicate), as well as make use of indoor location services. The core of the intelligent environment is a CBP agent. The CBP agent attends to clients requesting suggestions. The clients then use their personal agents installed on their handheld devices (PDA, mobile phone, etc.) to interact with the intelligent environment. The CBP agent proposes guidance suggestions depending on client preferences and the shops' capabilities.

The prototype implemented in this case study focused on the capacity to capture and classify SOAP messages in the shopping mall. To do this, the prototype incorporated the Traffic agents, the CBRMAS classifier agents and the XMLAnalyzer agent. These agents provide facilities to capture traffic, analyze the messages and provide a classification. The traffic agents were adapted to facilitate their download and installation in the handheld devices of the users in the shopping mall, in order to capture SOAM messages. The classifier agents were distributed between the PCs available for the experiment. Fig. 5 presents the architecture of the system used to implement and evaluate the preliminary prototype for the approach presented in this paper. As can be seen in Fig. 5, the approach presented in this paper to monitor and detect XML attacks is integrated within the previously existing multi-agent system in the shopping mall. Concretely, the security layer is located as an intermediate layer between the user's agents and the reasoning agents (planner and shop agents). In the inferior side of the image, it is possible to observe the different users that have access to the system using the Wi-Fi network of the mall or remote networks.

To evaluate our proposal it was necessary to determine the Web Services available in the previous existing multi-agent system (Corchado et al., 2009). As the solution was based in a multi-agent architecture, it is possible to easily obtain the roles that play the agents as well as the Web Services that will be offered in the system. This information is shown in Table 4.

In the case study, we have focused on the CBP Planner agent, and specifically in the planner role, because it is the agent with a greater number of services available and that deals with a high number of variability in the messages, since the number of elements in the messages varies depending on the request type. Nowadays, the system is installed and working in the mall and, although the results obtained are satisfactory, certain technical problems related to the Bluetooth network were detected. An average of 65 clients daily connects to the system through their handheld devices to make use of the services provided in the mall. The average number of queries processed by the CBP agent varies

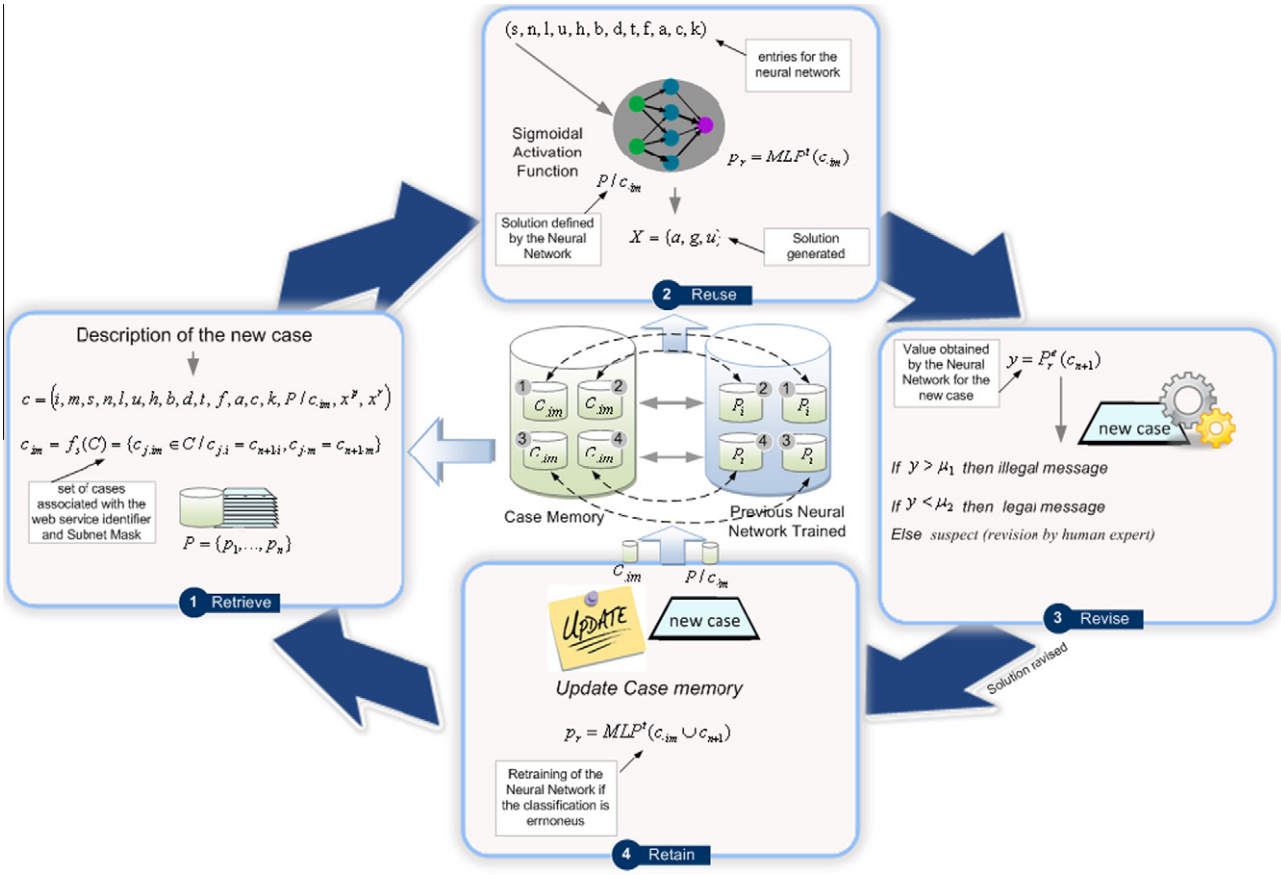

Fig. 4. Stages of the CBR cycle of a CBRMAS-L2 agent in the first phase of the classification mechanism.

between 12 and 16 per user, which generates an amount of 1040–1200 requests per day. The users' requests are mainly aimed at obtaining plans oriented to facilitate routes for shopping. The generation of routes requires to optimize the time available, to modify the plans in real-time and recover of previously stored plans. Another important group of requests are those oriented to close active sessions and to identify users. Finally, another significative group of requests (over 65 daily request) are those aimed at resolving incidents, complete surveys, etc.

All the previous mentioned services require an open platform, with Communications capabilities to connect clients, shoppers and directorship in the mall. All the requests from the users and processed in the multi-agent system are codified using SOAP messages. These SOAP messages can be sent from different devices, as PDAs, Smart Phone, laptops, etc. connected to the local network of the Shopping mall "as shown in Fig. 6" or via internet. The structure of the SOAP messages, for the Web Services offered by the CBP agent in the planner role, is defined through the information presented in Tables 5–9. This information represents the data associated to each of the plans.

A client profile contains information about a client's personal data (gender, economic level, postal code, number of children, and date of birth) and interests, and retail data (retail time and frequency, monthly profit – both business and product).

The global restrictions are applied on the whole plan and not on each of the individual shops.

The restrictions contain information about the time and money available Table 7.

A route is a list representing the suggestion presented to the client, available Table 8.

The route consists of various stages, each of which contains information such as the shop visited by the client, arrival time, the time spent in the shop, the products consumed by the client, and the next destination, available Table 9.

The structure of the SOAP messages for the Web Services of a CBP agent playing the planner role is shown in Table 10. Table 10 presents the fields required for the services as inputs. Furthermore, the fields of the outputs that the services provide are also shown.

An important information to take into account for each of the SOAP messages is the variability of the length of the messages. The variability in the number of elements and the level of nesting in the SOAP messages depends on the type of operation that executes the Web service. Generalizing for the Web Services to assess, the approximate size of the SOAP messages sent by users ranges varies from 30 kB to 950 kB. Once defined the structure of the SOAP messages for a CBP agent playing the planner role, the next step in the evaluation of the solution is to set a timeframe for monitoring the execution environment and to provide information with input data to the prototype, and a period of 10 days can be considered as significative to evaluate the system, due to the significant volume of SOAP messages generated during a working day. As the prototype requires a memory of cases containing previous experiences, the first 5 days were used to obtain initial information and the last 5 days to test the effectiveness of the solution. The procedure was developed by capturing the incoming SOAP messages independently of their origin, the local network or remote access via the Internet. This would facilitate later generation of the memories of cases taking into account the subnet masks. During the first 5 days a total of 1231 SOAP messages were obtained (after filtering

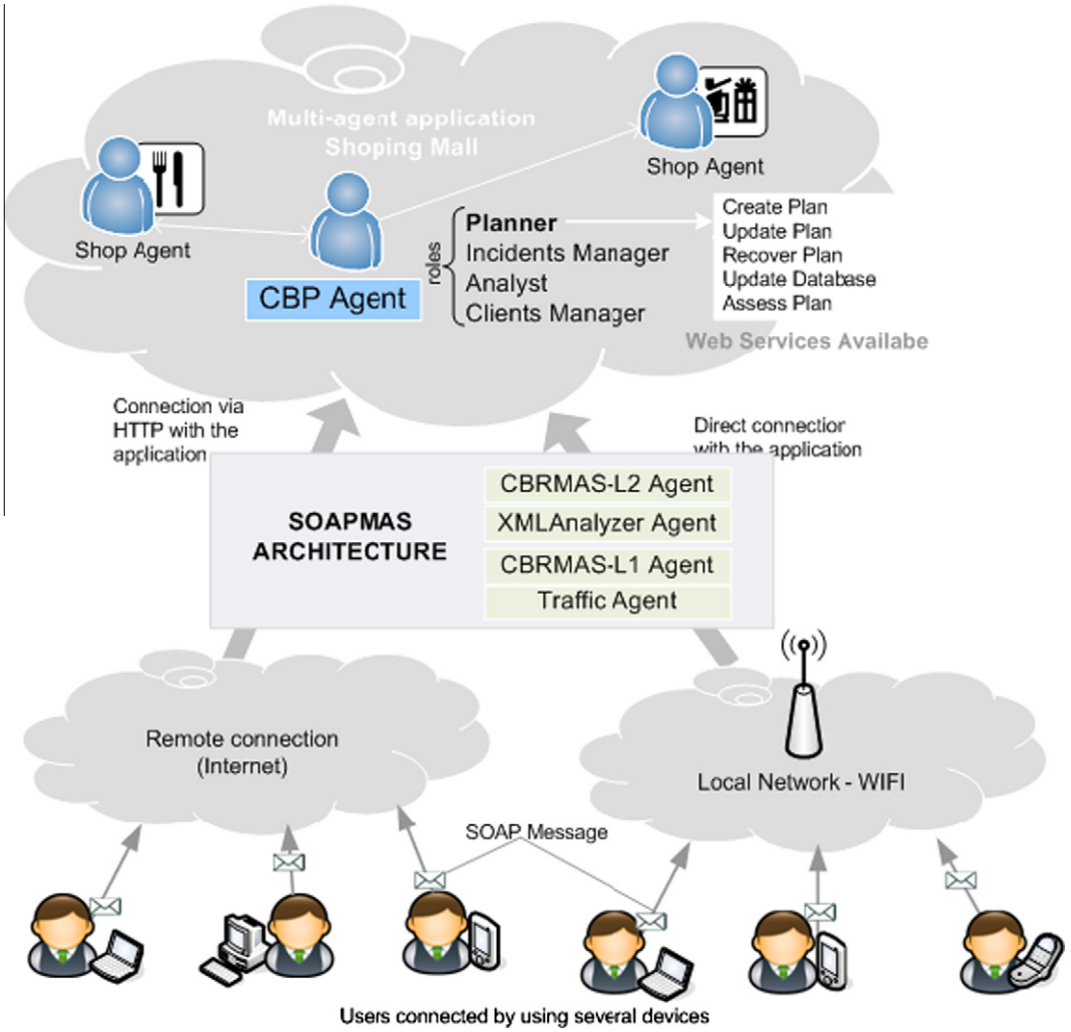

Fig. 5. Scenario of the previous multi-agent system installed in the shopping mall and the location of the S-MAS.

Table 4
Agents with the roles and Web Services available.

Planner	Incidents manager	Analyst	Clients manager
CBP Agent with the roles and Web Services available			
createPlan	addIncident	addUser	userList
modifyPlan	modifyIncident	analyzeSales	identifyUser
recoverPlan	deleteIncident	analyzePromotions	closeSesion
updateDatabase	queryIncident	addSurvey	
evaluatePlan	queryIncidents	analyzeSurvey	
		modifySurvey	
		querySurvey	
		querySurveys	
Communicator	Finder	Profile manager	
User Agent with the roles and Web Services available			
sendMessage	userLocation	addUser	
		deleteUser	
		modifyUser	
		queryUser	
		queryUsers	
Promotions manager	Store operator		
Shop agent with the roles and Web Services available			
addPromotion	addProduct		
deletePromotion	deleteProduct		
modifyPromotion	modifyStock		
queryPromotion	queryStock		
queryPromotions			

non-relevant messages). Table 11 details the distribution of the SOAP messages obtained for each of the Web Services provided by the CBP agent.

The messages retrieved were SOAP messages under normal operating conditions (legal messages). However, our approach requires malicious messages to identify when a message corresponds to an attack or not. To achieve this goal, we developed a set of illegal SOAP messages on the basis of the knowledge about the structure of the Web Services and SOAP messages used in the system. A set of 325 maliciously SOAP messages were generated, obtaining a total of 1556 SOAP messages. The distribution of the maliciously SOAP messages is presented in Table 12.

With this initial information it was possible to generate the memories of cases for each of the CBRMAS agents used in the phases of the classification mechanism. Once the cases memories were generated, the evaluation was carried out in the following 5 days. At this stage, we consider a specific number of users. The users selected for the evaluation sent queries during 5 days that were captured and analyzed by the classification mechanism. The queries of the users came both from the local network and form the Internet. As the clients in the mall submit legal queries, we decided to introduce malicious SOAP messages in order to check the classification mechanism. These malicious SOAP messages were launched form the local network and form different remote Internet locations (different networks). During the 5 days of the

Fig. 6. Example of web service accessible form a handheld device.

Table 5
Problem description structure.

ProblemDescription	Object type
Case Id	Integer
Initial Location	Coordinate
Client Profile	ClientProfile
Client Preferences	ArrayList of ProductPreference
Restrictions	ArrayList of restriction

Table 6
Product preference structure.

ProductPreference	Object type
MinimumPrice	Float
MaximumPrice	Float
StartTime	Date
FinishTime	Date
Product type	Integer
Shop type	Integer

Table 7
Planning restrictions.

Restrictions	Object type
TotalTime	Time
TotalMoney	Float

Table 8
Plan structure.

Plan	Object type
CaseId	Integer
route	Route
Quality	Float

tests, a total number of 1065 legal SOAP messages and 300 malicious SOAP messages were processed, which makes a total of 1365 messages.

Table 9
Route definition for a guidance suggestion.

Route	Object type
shop	Shop
ArrivalTime	Time
ServiceTime	Time
RetailProducts	ArrayList of product
NextShop	Route

Table 10
Summary of the structure of the SOAP messages, taking into account the parameters used as inputs for the Web Services and the parameters used as outputs.

Web Service	Input-Web Service	Output-Web Service
createPlan	Tables 5–7	
updatePlan	Tables 5–8	Tables 8, 9
recoverPlan	Tables 5–7	Tables 8, 9
updateDatabase	Tables 8, 9	
assessPlan	Tables 8, 9	

Finally, to conclude the description of the case study some technical aspects of equipment used to conduct the tests are provided. These aspects are an influential factor in the results obtained, since the performance of the system is a critical factor when assessing this type of approach. The prototype, and more specifically the mechanism of classification was tested using two standard PC connected via a 100 Mbps Ethernet network, using a physical switch which in turn was connected to the local network in the mall to capture the SOAP messages sent by users. Each PC was an HP Pavilion Intel Core 2 Duo E7200 with 4 GB RAM. The tasks for the classification mechanism were distributed among the two PC. Next section presents the results and the conclusions obtained.

6. Results and conclusions

SOAP messages used for communication in Web Services environments are vulnerable to DoS attacks. A DoS attack launched on a Web Services environment is a potential threat and can severely compromise the availability of the Web Services. A new approach is presented in this article based on a new hierarchical multi-agent distributed architecture, S-MAS, for blocking malicious SOAP messages. S-MAS, unlike the centralized solutions (Bebawy et al., 2005; Chonka et al., 2009; Gruschka & Luttenberger, 2006; Im & Song, 2005; Loh et al., 2006; Padmanabhuni et al., 2006; Srivatsa et al., 2008; Wang, 2006; Ye, 2008; Yee et al., 2007), is an adaptive approach that combines the advantages of multi-agent systems, such as autonomy and distributed problem solving (Corchado et al., 2008), with the adaptation and learning capabilities of CBR systems (Corchado & Laza, 2003; Corchado, Laza, Borrajo, Luis, & ValiÑo, 2003). Moreover, the user of decision trees and neural networks provides prediction and classification abilities, and their combination improves the overall functioning of the system.

The core of the architecture presented in this paper is the two-phases classification mechanism specifically designed to analyze

Table 11
Distribution of the SOAP messages captured by each of the web services of the planner role.

Web Service	Total messages
createPlan	185
updatePlan	326
recoverPlan	441
updateDatabase	123
assessPlan	156

Table 12
Distribution of the total of malicious SOAP messages.

Web Service	Mensajes Maliciosos
createPlan	110
updatePlan	35
recoverPlan	85
updateDatabase	73
assessPlan	22

and classify the SOAP messages. The two-phases mechanism provides a hybrid alternative in which a first classifier filters the messages with a low-resource consumption and provides an initial classification. The second classifier is only executed when the first classifier could not provide a reliable result. The second classifier provides a high reliable classification, but with high-resource and time consumption and uses as input data the elements obtained from the SOAP messages after a syntactic analysis of the structure and XML content of the messages.

The solution presented in this paper provides a new alternative for dealing with DoS attacks in Web Services environments, even when not overlooked penalizing response time required to execute the second phase of the classification mechanism. However, this penalty in response time is offset by the success of the solution because the procedure implemented in this latest phase of the classification mechanism allows a robust classification. Another important point to note in our solution is the ability to provide distributed and hierarchical capabilities. The distributed approach and the hierarchical design of the architecture helps to distribute tasks and to clarify how these tasks are assigned according to the roles of the different agents in each layer of the architecture, not forgetting the added capabilities offered by hierarchical structure for easy errors recovery. When an agent in a concrete layer is compromised, the agent is eliminated and a new instance is created without affecting the rest of agents in the same layer or other layers of the architecture. Finally, a classification into two phases allows automatic solving of most of the incoming SOAP messages, reducing the possible intervention of a human expert to solve the final classification.

To check the validity of the proposed model, we elaborated a series of tests. The results obtained were promising, which allows us to conclude that S-MAS can be considered a good alternative for detecting and blocking of malicious SOAP message. The tests were conducted as follows: The first step was to generate the cases memories for the CBR engine of the CBRSOAP agents in every phase of the classification mechanism. Memories are made of cases involving the legal messages as well as maliciously messages. The cases memories were generated from the messages received in the first 5 days of monitoring the application installed in the mall, as explained in the case study. The next step was to test the prototype with a set of test performed with data obtained during the last 5 days of monitoring. The tests allow us to evaluate the global efficiency of the architecture by comparing different meaningful parameters before and after the implementation of the system in the test environment. The following paragraphs describe the experiments and discuss the conclusions obtained.

The first element to evaluate is the response time of S-MAS approach by analyzing the response time depending on the size of the messages. For this test we took a set of 200 messages of varying sizes and were entered into the system during the five testing days. Before entering the data into the system the information stored for those cases was deleted. As the system evolved during the days of testing, the response time is improved, but the message size directly affects the response time. The results obtained for this test are shown in Fig. 7. In Fig. 7, the *X*-axis shows the size of the SOAP messages (kB), while the *Y*-axis shows the response time in milliseconds (ms).

The next element to evaluate is the percentage of false positives and false negatives generated by S-MAS, which are shown in Fig. 8. The results obtained after the tests and shown in Fig. 8 show that each of the radios represent the day of the test, and the breadth of the graph represents the percentage detected for the false positives and false negatives. In parentheses is represented the time lapse (in days) from the construction of the cases memories and the total number of cases stored.

To validate the evolution of the system, we proceeded to check the error rate as the cases were registered in the system. In Fig. 9, it is possible to see the result obtained, and the decreasing trend in the system. The *X*-axis represents the number of cases and the *Y*-axis the percentage of errors found for the number of cases. It is clear that a large number the pattern of training improves the percentage of prediction. As we are working with CBR systems, which depend on a larger amount of data stored in the cases memory for each user, the percentage of success in the prediction increased. CBR systems need to draw from initial information (past experiences) in order to generalize efficient results.

As the number of cases in the cases memory of the CBRMAS-L1 agent increases, the number of times that it is necessary to execute a CBR cycle of the CBRMAS-L2 agent decreases, and this fact produces a reduction of the total execution time of the system. In Fig. 10 can be seen the percentage of execution for each of the CBRMAS agents along the 5 days following to the initiation of the system, and the average time in milliseconds (ms) obtained for the

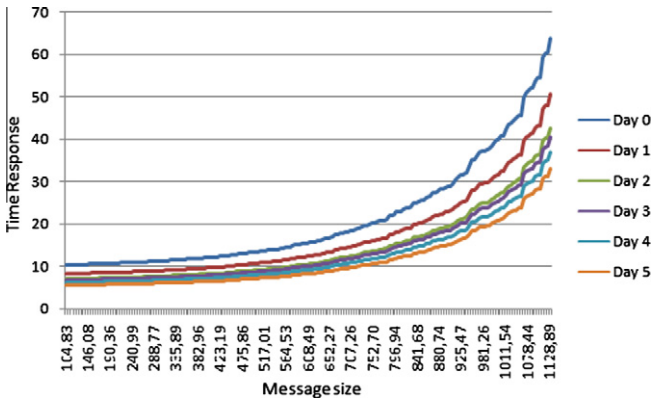

Fig. 7. Evaluation of the response time depending on the size of the messages for the five testing days.

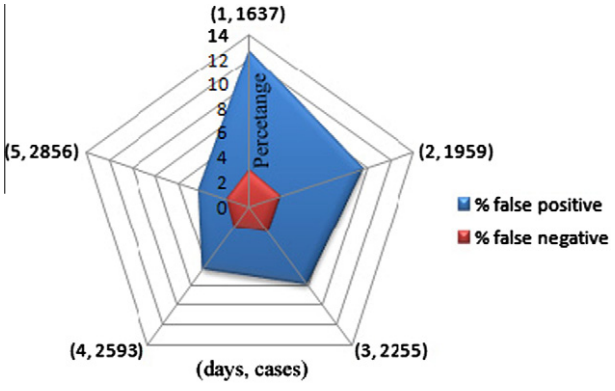

Fig. 8. Percentage of false positives and false negatives detected in the system.

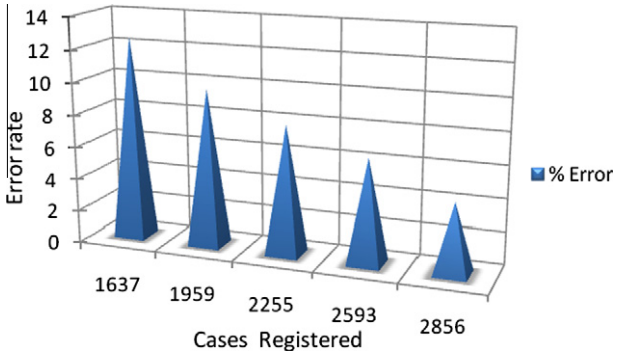

Fig. 9. Error rate depending on the number of registered cases.

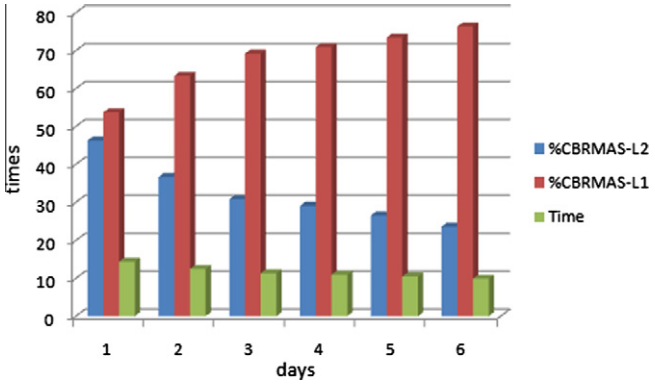

Fig. 10. Percentage of execution for each of the CBRMAS agents along the five testing days and average execution time obtained for the classification of services.

execution of the classification of the services. The *X*-axis shows the initial and subsequent results obtained during the five testing days and the *Y*-axis represents the percentage of use of each of the CBRMAS agents, and the evolution of the average time. As can be seen, the average time decreased as the percentage of the use of the CBRMAS-L1 increased.

Finally, we examined the percentage of times that each of the CBRMAS agents were executed along the five days of testing. Fig. 11 shows the results obtained for each of the CBRMAS agents, and, as can be seen, the percentage of times for both agents decreases significantly. The percentage of times that the decision tree is rebuilt is always higher than the neural network, since the neural network is only rebuilt when the second CBR is invoked and an error occurs, and the decision tree is always rebuilt when the second CBR is invoked. This significantly improves the performance of the system by decreasing the number of times required to train the neural network.

The architecture presented in this paper provides a novel strategy for blocking malicious SOAP message. S-MAS evolved from the previous SQLMAS architecture (Bajo et al., 2008; Pinzón et al., 2008) that was aimed at detecting and blocking SQL injection attacks. In this sense, S-MAS has improved the design of the SQLMAS architecture facilitating a distributed perspective, as well as the distribution of services strategies. Moreover, S-MAS provides a novel classification method, since implements a two-phases classification, which notably improves the general performance of the system and reduces the response time with respect to SQLMAS. Finally, the classification strategies have been adapted to the needs of the XML injection problem. As shown in this paper, S-MAS proposes a new distributed perspective for detecting and preventing attacks and improves the functionalities offered by the existing approaches. The results are promising and allow us to conclude that

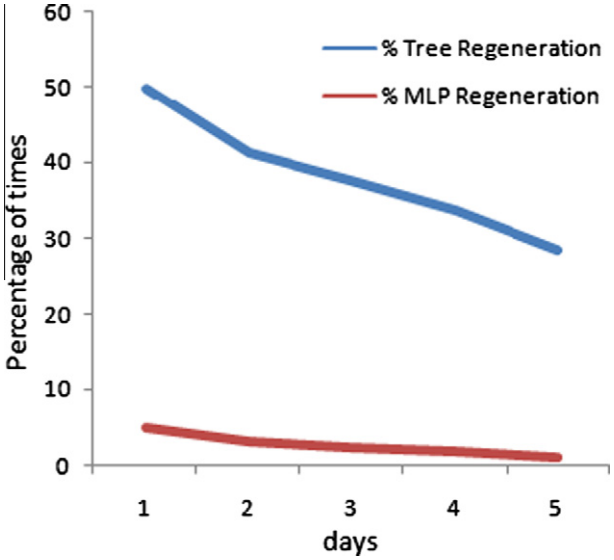

Fig. 11. Percentage of times that each of the CBRMAS agents are executed along the five testing days.

the S-MAS architecture considerably can be considered as a solid alternative to prevent and detect DoS attacks in web service environments. However, there is still much work to do, especially checking the validity of our architecture in heterogeneous real environments. These are our next challenges.

Acknowledgements

This development has been partially supported by the Spanish Ministry of Science Project TIN2006-14630-C03-03 and The Professional Excellence Program 2006–2010 IFARHU-SENACYT-Panama.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications, 7*, 39–59.

Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., & Dixon, B. (2004). *Web services secure conversation language (WS-SecureConversation)*.

Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della, G., & Dixon, B. (2005). *Web services trust language (WS-Trust)*.

Bajo, J., Corchado, J. M., Pinzón, C., Paz, Y. D., & Pérez-Lancho, B. (2008). SCMAS: A distributed hierarchical multi-agent architecture for blocking attacks to databases. *International Journal of Innovative Computing, Information and Control*.

Bebawy, R., Sabry, H., El-Kassas, S., Hanna, Y., & Youssef, Y. (2005). *Nedgty: Web services firewall*.

Brownell, D. (Ed.) (2002). SAX2. O'Reilly & Associates, Inc.

Carrascosa, C., Bajo, J., Julian, V., Corchado, J. M., & Botti, V. (2008). Hybrid multi-agent architecture as a real-time problem-solving model. *Expert Systems with Applications, 34*, 2–17.

Corchado, J. M., Bajo, J., & Abraham, A. (2008). GerAmi: Improving healthcare delivery in geriatric residences. *Intelligent Systems, IEEE, 23*, 19–25.

Corchado, J. M., Bajo, J., DePaz, J. F., & Rodríguez, S. (2009). An execution time neural-CBR guidance assistant. *Neurocomputing*.

Corchado, J. M., & Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems, 18*, 1227–1241.

Corchado, J. M., Laza, R., Borrajo, L., Luis, J. C. Y. A. D., & ValiÑo, M. (2003). Increasing the autonomy of deliberative agents with a case-based reasoning system. *International Journal of Computational Intelligence and Applications, 3*, 101–118.

Chonka, A., Zhou, W., & Xiang, Y. (2009). *Defending grid web services from XDoS attacks by SOTA*.

Fujii, K. (2000). *Jpcap – a network packet capture library for applications written in Java*. <http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html>.

Della-Libera, G., Hallam-Baker, P., Hondo, M., Janczuk, T., Kaler, C., & Maruyama, H. (2005). *Web services security policy language version 1.0 (WS-SecurityPolicy)*.

Gallagher, M., & Downs, T. (2003). Visualization of learning in multilayer perceptron networks using principal component analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 33*, 28–34.

Gruschka, N., & Luttenberger, N. (2006). *Protecting web services from DoS attacks by SOAP message validation*.

Im, E. G. & Song, Y. H. (2005). An adaptive approach to handle DoS attack for web services. In S. B. Heidelberg (Ed.).

Jensen, M., Gruschka, N., Herkenhoner, R., & Luttenberger, N. (2007). SOA and web services: New technologies, new standards – new attacks. In *Fifth European conference on web services*.

Laza, R., Pavó, N. R., & Corchado, J. M. (2003). A reasoning model for CBR_BDI agents using an adaptable fuzzy inference system. In R. Conejo, M. Urretavizcaya, & J.-L. P. De-la Cruz (Eds.). Springer.

Lecun, Y., Bottou, L., Orr, G. B., & MÜller, K. R. (1998). *Efficient BackProp*. *Neural networks: Tricks of the trade*. Berlin/Heidelberg: Springer.

Loh, Y.-S., Yau, W.-C., Wong, C.-T., & Ho, W.-C. (2006). Design and implementation of an XML Firewall. In *International conference on computational intelligence and security* (Vol. 2, pp. 1147–1150).

OASIS (2004). Web services security: SOAP message security 1.1 (WS-Security 2004).

Padmanabhuni, S., Singh, V., Kumar, K. M. S. & Chatterjee, A. (2006). *Preventing service oriented denial of service (PreSODoS): A proposed approach*.

PinzóN, C., Paz, Y. D., & Bajo, J. (2008). A multiagent based strategy for detecting attacks in databases in a distributed mode. In J. M. Corchado, S. Rodríguez, J. Llinas, J. M. Molina, *International symposium on distributed computing and artificial intelligence (DCAI2008), Salamanca, Spain, Berlin*.

Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A., & Zamboni, D. (1997). *Analysis of a denial of service attack on TCP*. Washington, DC, USA: IEEE Computer Society.

Srivatsa, M., Iyengar, A., Yin, J., & Liu, L. (2008). *Mitigating application-level denial of service attacks on Web servers: A client-transparent approach*. ACM.

Wang, J. (2006). *Defending against denial of web services using sessions*.

Ye, X. (2008). *Countering DDoS and XDoS attacks against web services*.

Yee, C. G., Shin, W. H., & Rao, G. S. V. R. K. (2007). An adaptive intrusion detection and prevention (ID/IP) framework for web services. In *International conference on convergence information technology (ICCIT '07)*. Washington, DC, USA: IEEE Computer Society.