Multi-domain fault management architecture based on a shared ontology-based knowledge plane

Alfonso Castro, Beatriz Fuentes, Jose A. Lozano
Autonomics System Department
Telefónica Investigación y Desarrollo, TID
Madrid, Spain
{acast, fuentes, jal}@tid.es

Begoña Costales
Operations Department
Polar
Madrid, Spain
mcostales@polar.es

Victor Villagrá
Telematics Department (DIT)
Technical University of Madrid (UPM).
Madrid, Spain
villagra@dit.upm.es

Abstract—**Nowadays the number of services whose functionalities need to adapt across heterogeneous networks with different technological and administrative domains has substantially increased. Each domain involves different management procedures; therefore the comprehensive management of multi-domain services presents serious problems.**

This paper is focused in fault management aspects and presents a novel management architecture for multi-domain environments. The architecture is based on a distributed set of agents, using semantic techniques. A Shared Knowledge Plane has been implemented to ensure communication between agents.

Keywords-component; Network Management; multi-domain; semantic; Shared Knowledge Plane

I. INTRODUCTION

In the last years, advances in technology have facilitated the emergence of new types of access networks, e.g. ADSL, Fiber, UMTS, WIFI or 3G. The number of network domains associated with services or customers could change at any time. Accordingly, the management solutions related to this new approach requires a greater dynamism than conventional solutions. Users could receive a service through UMTS access in their car and continue receiving the same service through ADSL access when they arrive home.

These new approaches present as significant management challenge: the integrated management for successful delivery of high quality services, especially when services run across heterogeneous networks. Current management systems are unable to cope with the complexity, heterogeneity and automation required by this vision. This paper seeks to explore the challenges in managing telecommunication network across different technological domains, and in particular fault management.

Classic fault management systems are associated with their domains: each network has its own fault management system that detects, logs, notifies users of network problems to keep the network running effectively and determinate the suitable actions to be performed in order to correct faults encountered in the network. Many commercial solutions cover this mono-domain situation. The current solutions for the multi-domain environment are based on creating an "umbrella system" that receives events and collects data from each domain to perform integrated fault management. These solutions are very static and have a monolithic data structure that defines the system operation. The addition of a new network domain is a difficult task and the correlation of events from different networks domains is a challenge.

Habitually, event occurrences are caused by problems (fault) in network elements, such as hardware or software failures, performance bottlenecks, configuration inconsistency, etc. An alarm consists of a notification of the occurrence of a specific event, which may or not represent an error. A single network problem may affect the operation of many elements and it may cause a great number of events. As a result, management systems are usually flooded by a large number of alarms when a few failures occur. This situation affects both mono-domain and multi-domain environments. Accordingly, alarm correlation is an important fault management process to address this problem in many network management systems. It involves not only information about the managed network, such as topology, configuration, but also knowledge about the events generated from network elements, i.e. format and meaning of each individual event, causal and temporal relationships among events and etcetera.

The work presented in this paper proposes an architecture based on distributed agents to develop fault management systems to be applied in a multi-domain environment. Semantic-based techniques are included in the development of these agents.

The addition of semantic descriptions provides a flexible definition of concepts associated to alarms from different domains, helping to solve the correlation problem. For example, an event of link down in the optical fibber domain generates some alarms which are different from those generated in the UMTS domain, but all of them could have the same treatment: ticket generation. This action is only associated with the meaning of the alarm and not with technological issues.

978-1-4244-8909-1/$26.00 ©2010 IEEE

This paper was peer reviewed at the direction of IEEE Communications Society by subject matter experts for publication in the CNSM 2010 proceedings.

Furthermore, the combination of a semantic description with formally-defined reasoning rules provides inference capability, which could be used in order to improve the event correlation through different domains.

The remainder of this paper is organized as follows. Section II shows the requirements associated with the multi-domain alarm management. Section III presents a set of existing solutions and its disadvantages for multi-domain management. Section IV explains how semantic techniques have been applied to design a new management architecture for network management in multi-domain environments based on a Shared Knowledge Plane (SKP) as the main element of the architecture, a fault ontology and a set of associated rules. Section V describes a proof of concept that validates the exposed principles and ideas; it consists of a couple of agents managing an ontology and a set of rules, that applies general treatment to alarms from different domains. Finally, section VI includes some conclusions and further work in this area.

II. EXISTING SOLUTIONS

Currently, there are a lot of commercial solutions for fault management, such as IBM Tivoli [1], HP Network Management Center [2] or Spectrum Service Availability [3]. These solutions are based on a hierarchical centralized architecture and need the overall knowledge of the environment (network topology, inventory…) in order to achieve optimal results. The addition of a new domain is complicated because it should incorporate the new description of the environment in the global image of the system.

Fault management has been a focus of research activities since the advent of modern communication systems. Most efforts are focused in the fault localization problem and the diagnostic and different techniques have been used including neuronal networks and decision tree [4, 5], probabilistic method [6], etc.

Recently, semantic model-based fault management solutions have been researched. [7] proposes a web service-based abstraction layer for home area network service composition and a semantic modeling, enriching the atomic services for fault management interfaces with semantic annotations from a fault management ontology. [8] presents a domain-ontology driven multi-agent based scheme for representing the knowledge of the Network Management System.

Some researchers have been studying the way to spread intelligence through the network. In [9], the Knowledge Plane is described as a distributed and decentralized construct within the network that gathers, aggregates, and manages information about network behavior and operation. [10] presents an abstract architecture for knowledge networks that addresses the key issues of how both physical contextual knowledge and social knowledge from the users can be used to create a knowledge space for supporting autonomic agents dealing with network elements and applications. Although both proposals are at the abstract level, they have been the foundations of more recent work, as [11], that applies this concept to a cross-layer architecture.

III. PROPOSED ARCHITECTURE

Now we present a proposal for alarm management in a multi-domain environment, applying semantic-based techniques. The suggested architecture is based on a set of distributed agents and a SKP. The knowledge plane is a distributed and decentralized construct that manages information about the fault environment, based on an ontology, and provides an integrated view to all agents.

A. *Distributed agents*

Each agent is associated with a network element in a specific domain and is devoted to alarm management. In the case of multi-domain, the behaviour of the different agents governing the involved domains will get the overall management. The global behaviour is defined by the composition of the individual behaviour of every agent.

In a similar way, each agent has attached a knowledge plane with the conceptual description of the domain and the rules that characterize the management activities. Each agent is aware of its domain, and uses the information described in the knowledge plane to infer the most appropriate actions to follow in each situation. The set of the knowledge planes of all the agents is the Shared Knowledge Plane. Therefore, the SKP can be seen as a set of non-disjoint sub-planes, one per domain.

Agents of the proposed architecture are implemented following the autonomic manager structure from the IBM MAPE-K [12] architecture, as shown in Fig. 1.

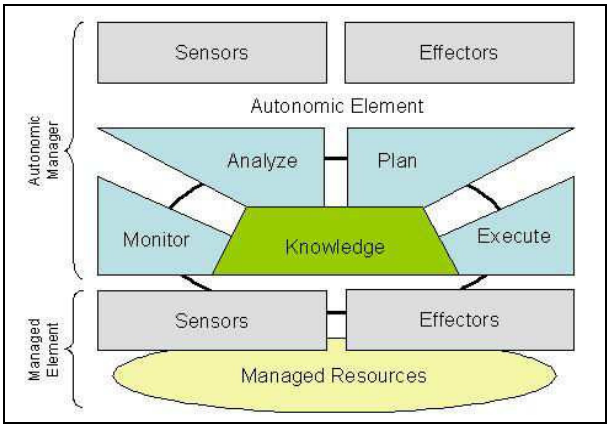

Figure 1. Agent structure MAPE-K (source IBM).

In the suggested architecture, Knowledge is the necessary description of the domain required by the agent, and it is implemented through an ontology. The ontology is used to make a conceptual description both of particular parts associated with each domain and common ones. An ontology for a fault management system is presented later.

Agent Monitor module captures alarms from its managed resource, the network element and modifies the SKP incorporating them as new instances. The Analysis Module is responsible for the reasoning function. The alarms are then analysed according to given reasoning rules that allow the inference process.

This inference process takes place to find out the appropriate actions. Once the inference process has been completed, the agent plans how to carry out the appropriate management actions. These actions may involve an operation that affects the network elements, but also includes the updating of the domain-specific knowledge plane, that in turn means the modification of the global SKP. Accordingly, the actions planned by an agent can influence the behaviour of other agent. In this way, the agents create the global behaviour.

Summarizing, the agent is presented as a software component that is allocated in a given domain and has the capability to infer –using information taken from the context– how to plan concrete actions upon reception of events coming from different domains. (See Fig2).

B. Shared knowledge plane for alarm management

The fundamental element of the above architecture is the SKP, which contains the semantic model, the set of instances and the reasoning rules that govern the behaviour of agents. The fact that the domain description and the management rules are enclosed within the ontology reduces the complexity of the agents. As it was introduced above, SKPs are composed by the union of the entire knowledge plane from each agent. This union is not disjoint, so the modification of the part associated to an agent could affect the behaviour of other agents.

The first step is the design of a conceptual model describing the domain, and its associated ontology. OWL [13] [14] (Ontology Web Language) has been chosen as implementation language. The use of an ontology language to define the management information offers advantages such as the ability to use reasoning tools working on ontologies (e.g., inference engines used in artificial intelligence).

The second step consists of the implementation of the rules governing the system behaviour, using SWRL [15] (Semantic Web Rule Language) that extends the abstract syntax of OWL rules to include conditional rules into the ontology language.

SKP is used by the agents to communicate between them, perceiving the behaviour of each other. Technically, the agents read or update information ontologies published in a web server with a mechanism similar to web 2.0 Wikis. This functionality is based on the work performed in [16]

In the traditional umbrella systems, alarm correlation is associated with the fulfilment of a rule that includes conditions from different domains. In the proposed architecture, this rule is split into several rules in order to simplify the multi-domain alarm management. If an alarm is only associated with one domain, it modifies the Knowledge Plane associated with this domain and the common part.

The incorporation of a new domain to the system involves the modification of the semantic model or the inclusion of new rules. It does not imply any change of code in the software process. In addition, if the model has been correctly defined, the modification will be minor, since concepts should be similar.

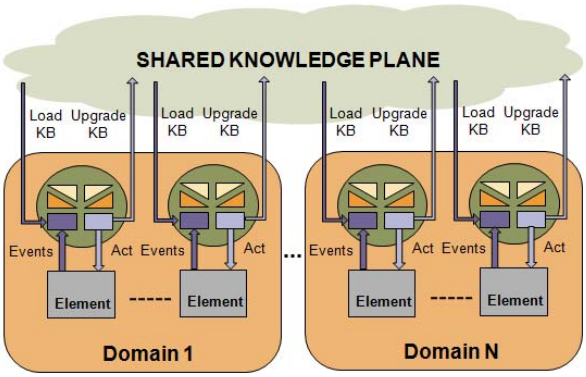

Figure 2. Dritributed agents with shared knowlegde plane

C. Ontology

In this section, we present the proposed ontology for the fault management. The concepts of this ontology are based on the fault management foundation which comprises the set of functions aiming to detect, isolate and correct a malfunctioning in telecommunication networks and services. One of its fundamental blocks is the alarm management system. An alarm can be defined as an indication to draw attention on a given condition that may have possible negative repercussions on the network element or service being supervised.

Figure 3 presents these concepts and relationships in this global shared ontology regarding alarm management

Element (network resource) has associated different management aspects. Network resources can notify their state, malfunction and some information about their normal operation –e.g. occupancy level and throughput.

Notification refers to the information issued by an element in connection with an event occurred in the network. A notification may be an alarm, a change of state or other spontaneous event. Upon reception of notifications and after an analysis of them, a set of actions are planned and implemented in order to correct faults and failures.

Action is a set of tasks carried out after having analysed a notification. These actions include those aimed to correct a fault or failure of an element. Specialized actions are: **Diagnosis** (identification of the root cause that leads to failure or fault on an element or service), **Repair**, (corrective action taken on an element or system), **Communication** to an External system, or **Storage** (management of the information associated to a fault or failure of an element), and **Execute**.

Alarm: Notification to draw attention to a condition that can have an immediate or potential negative impact on the state of the element or service being monitored.

Alarm description: A set of generic attributes related to the alarm, for example: identity, date/time of the alarm activation/cessation, severity, type, priority, state (active, ceased, under reparation). Alarms in each technological domain may have some attributes different from those described here.

Cause description: A set of particular attributes that describe the cause of the alarm: identity of the causing element or service, type of failure, date/time of the failure or probable cause (information about the origin of the alarm).

Change of state. Notification received when a change in the state of an element or service occurs

Information: Notification used as an item in a report.

Counter: After the arrival of a notification and in order to conduct the necessary analysis, a specific counter can be created, increased or deleted. This counter measures the number of notifications of the same class or with the same value for any attributes that have been received.

Repetition: Specify counter characteristics: maximum number of repetition (threshold) in a time interval.

Timer: After the arrival of a notification, it may be necessary to activate a timer in order to perform actions related to persistence, accumulation or repetition of a concrete alarm or a specific type of alarm for a period of time.

Persistence: Specify the characteristics of the timer: period of activity and value in and out of it – i.e., from 9am to 6 pm put a timer of 2 minutes, and 5 minutes out of this interval.

D. Reasoning rules

In the semantic description of fault / alarm domain, it is necessary to define the set of rules that will govern the implementation of preventive or corrective actions related to the alarm management system. A representative set of rules has been chosen, determining the actions to be carried out after the analysis of the alarms within a generic fault management system. These rules apply to notifications coming from all domains.

The selected rules cover the following functionalities:

- Activation of an irrelevant alarm. In this case the alarm is filtered and not included in the knowledge base.
- Activation of an alarm that needs to be addressed. When one of these alarms is received, it is stored in the repository and persistence timers are created in order to check the evolution of the possible failure in the timeline (i.e., if the alarm persists for a certain period of time, the unavailability of a network element or service could be determined).
- Cessation of an alarm. If it corresponds to an active alarm, the alarm is immediately deleted from the repository and from the KB. On the other hand, if the alarm was inactive, the removal is delayed through the use of a timer.
- Duplicated activation. Only the first activation of an alarm is taken into account.
- Triggering of an unavailability alarm. Upon the expiration of the persistence timeout the system calculate the unavailability level (<50%, 50%, 75%, 100%) in base on topological information included in AlarmDescription and CauseDescription. If it is greater than or equal to 50% the execution of the corresponding task is ordered to an external system. Detection of unavailability levels less than 100% can be used to avoid any faults that may follow.
- Triggering of a repetition alarm. The successive activations / cessations of an alarm are accounted. If they exceed a threshold in a given time the execution of a task is ordered.

IV. PROOF OF CONCEPT

The main purpose of this proof of concept is to ensure the correct inference of the appropriate corrective actions based on the received alarms, using a semantic model and an associated set of rules. In this way, we validate if the model fulfils the basic functionalities of an alarm management system. In addition, we have measured the time spent in the execution of every rule in order to evaluate if the strict time execution requirements of any alarm system can be accomplished.

This proof of concept shows the way in which the agents work and its interaction with the share knowledge plane (see Fig 4). Therefore, the above described model has been applied to a real system that receives alarms from switching exchanges of various technologies. These notifications from different domains present the same format with some common fields to all the domains and some others specific for each one.

Our prototype has two agents that apply a general processing to all the notifications taken into account, both common and specific fields. The prototype has been tested in a lab environment integrated in an operative alarm system receiving real notifications.

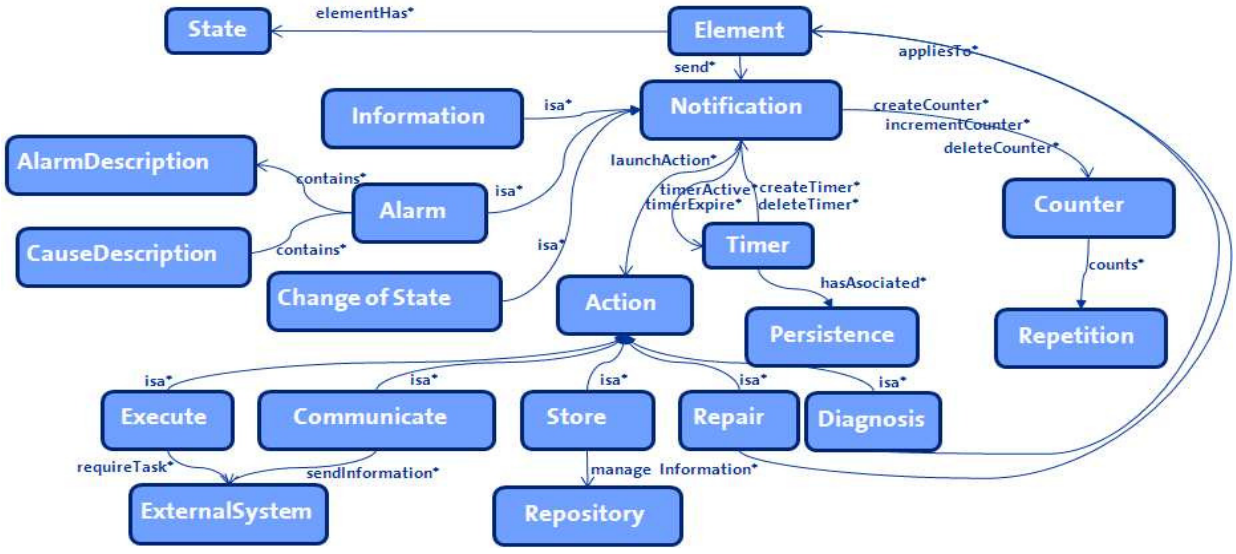

Figure 3. Fault ontology.

These agents interact through the Shared Knowledge Plane. Each agent adds to the SKP, jointly with common alarm information as activation/cessation, some topology information included in the alarm content, for example, the route where the network element sending the alarm is located. So, topology information is available on the SKP to agents of all domains and anyone of them can detect the unavailability of a route where there are elements of different domains.

For example, the agent of domain A receives an alarm about the failure of a network element in a determinate route. Then, the agent modifies the KB into the common part, creating an instance of the alarm in which the failure type, the element identity and the route are indicated. At this point, the agent of the domain B receives an alarm about the failure of a network element in the same route, analyses the amount of elements of the route in failure and evaluates the unavailability level.

Knowledge plane has been described by means of an Ontology written in OWL1 language. We have used Protégé 3.4 [17] to edit the ontology and behavior rules which have been written in SWRL (Semantic Web Rule Language).

Each agent was developed in java, and it uses the Java library Jena 2.5 [18] to perform the parsing and manipulation of the ontology information together with the Bossam reasoning engine to execute the rules. Bossam is not integrated in Jena, for that reason the agent needs to write the KB in an owl file that Bossam reads to build its KB with the same content but with its own structure.

Each agent receives the alarms from the network and includes the corresponding Alarm, AlarmDescription and CauseDescription in the SKP. Afterwards it uses the reasoning engine to execute the rules and infer the actions to carry out. Fig. 5 shows the execution process of the proof of concept which is split in the following steps:

- Definition of the semantic model (OWL+SWRL) of the shared knowledge plane and generation of the OWL text file using protégé.
- Creation of the Knowledge Base (KB) in the SKP based on the OWL+SWRL model.
- Update the KB in the SKP with the received alarms from any domain.
- Storage of this KB in an owl file in each agent.
- Bossam engine reads the owl file and builds its own KB in the agents
- If it is necessary, execution of an inference cycle in the agents.
- Modification of the Bossam´s KB following the reasoning results
- Planning and ordering of the actions to be carried out
- Update the KB in the SKP ane with the reasoning results
- Each agent load the new KB

The complete test involves two test environments. A local environment in a Windows PC with simulated notifications was used to check the fulfillment of every rule, and a unix lab environment receiving real notifications from the network where we have evaluated the accomplishment of the time execution requirement.

First tests showed that SWRL rules covered the functionality. However, time measurements made in this environment with an empty KB were disappointing; the following table shows these measurements

TABLE I. TIME MEASURES RULE FUNCTIONALITY

Rule	Time (ms)
Activation of an irrelevant alarm	33797
Cesation of an alarm that is not active	32297 30703
Cessation of an active alarm	35427
Activation of an alarm that need to be addressed	35328
Duplicated activation	35906
Triggering of an unavailability alarm (100%)	33234
Triggering of an unavailability alarm (75%)	31218
Triggering of an unavailability alarm (50%)	31813
Triggering of a repetition alarm (3 repetitions)	32562 32641 32172 31203

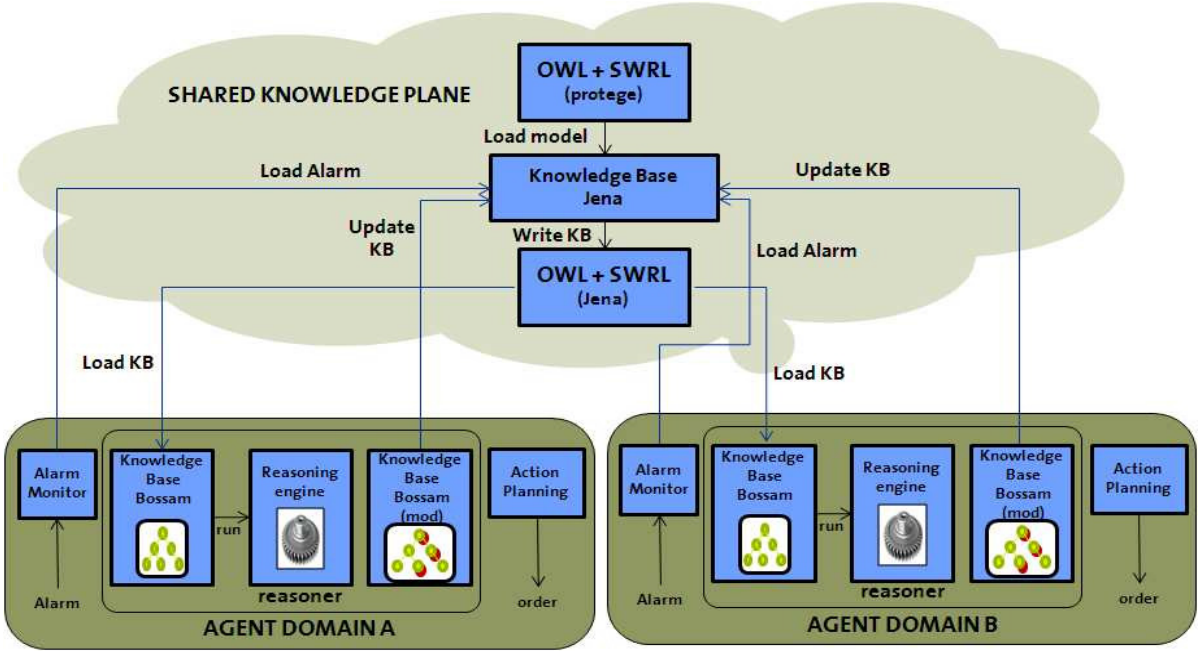

Figure 4. Proof of concept schema.

The time needed to write/read Bossam's owl file is an important part of time execution. In order to avoid this step we attempted to use Pellet reasoner which has a direct interface with Jena. But in this case the time response was even worse, 1208 s for the activation of an irrelevant alarm, comparing to the 34s needed by Bossam. Furthermore, Pellet does not support the complete set of SWRL built-ins needed to implement the rules and consequently it cannot cover the whole functionality.

For the second test we have moved ontology and applications to a real environment and measured time to receive and process alarms. In these conditions the system received 300 alarms in 13 seconds, but it needs 57 seconds to process just the first alarm, and execution time increases according the size of the KB.

These results show the lack of efficient implementation of the reasoners; therefore, the model does not fulfill time requirements. This problem can be overcome with a notable improvement in the reasoning algorithmic (like the hypertableaux in HermiT) [19] and with the structural changes in OWL 2. All of this can make the efficiency no longer a valid excuse not to use ontologies.

V. CONCLUSIONES AND FUTURE WORKS

This paper has presented a novel approach to the fault network management for multi-domain environments, based on distributed agent architecture. The fundamental element of this architecture is the SKP, described as an ontology, which is a semantic description of the fault domain. The agents are allocated in the network elements of each domain and all of them share knowledge about their domain. Agents receive alarms, analyse them and infer the suitable actions using a rule-based mechanism, and upgrade the knowledge plane.

The incorporation of a new network domain to the fault management will be performed by adding its associated knowledge plane to the SKP. This could be done by upgrading the ontology model with the new concept, relations and attributes from the new domain that were not described before. Semantics will provide flexibility to define the concepts associated with the alarms in different domains; therefore, the first problem of correlation of events from different domains is addressed. Moreover alarm correlation from different domains can be performed by the rules associated with the ontology.

This work will be taken as an initial point in order to define new service management architectures. In this future architecture each agent is associated with a network or service element in a specific domain and it is devoted to a concrete management aspect, such as provisioning, monitoring, guaranteeing the QoS, SLA management or fault recovery of those instances.

REFERENCES

[1] Tivoli IBM, http://www-01.ibm.com/software/tivoli/, accessed: 28 may. 2010

[2] HP Network Management Center, https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119_4000_100__, accessed: 28 may. 2010

[3] CA Inc, http://www.ca.com/files/IndustryAnalystReports/spectrumserviceassurance_230577.pdf, accesed: 24 ago 2010

[4] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in Computer Networks". [Elsevier] Science of Computer Programming, S.I. on Network and System Administration, 2004

[5] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer. "Failure diagnosis using decision trees". In Proc. Intl. Conference on Autonomic Computing, New York, NY, 2004

[6] Burgess M "Probabilistic anomaly detection in distributed computer networks" Science of Computer Programming vol. 60, no1,pp. 1–26, 2006

[7] Z. Etzioni, J. Keeney, R.Brennan, and D. Lewis, "Supporting composite smart home services with semantic fault management" to appear in Proceedigns of the 5th International Symposium on Smart Home (SH 2010) at FutureTech 2010, Busan, Korea, 21-23 May 2010

[8] S. Abar et al., "Exploting domain ontologies and intelligent agents: An automante network management support paradigm" LNCS Vol 3961, 2006

[9] D. Clark et al., "A knowledge plane for the Internet", Proceedings of the 2003 ACM SIGCOMM Conference, Karlsruhe,ACM Press, 2003

[10] M. Mulvenna, F. Sambonelli, "Knowledge networks: the nervous system of an autonomic communication infrastructure" 2nd IFIP Workshop on Autonomic Communication, LNCS No 3854, 2006

[11] M.A. Razzaque, S. Dobson and P. Nixon "A cross-layer architecture for autonomic communication" , In proceedings of Int'l Workshop on Autonomic Communications, Paris, Sep 2006.

[12] IBM, "Autonomic computing: IBM's Perspective on the State of Information Technology", http://researcweb.watson.ibm.com/autonomic/manifesto/autonomic_computing, accessed: 28 may. 2010.

[13] D. L. McGuinness, F. van Harmelen, "OWL web ontology language overview" Acknowledged W3C Member Submission. May 21, 2004.

[14] Web Ontology Language (OWL) http://www.w3.org/2004/OWL/, accessed: 28 may. 2010

[15] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., "SWRL: A semantic web rule language combining OWL and RuleML" W3C Recommendation, 10 February 2004 http://www.w3.org/Submission/SWRL/, accessed: 28 may. 2010

[16] J.M. Gonzalez, J.A. Lozano, A. Castro, "Autonomic system administration. A Testbed on Autonomics", Procedings of Fifth International Conference on Autonomic and Autonomous Systems, pag 117-122, 2009

[17] Protegé, available at http://protege.stanford.edu, accessed: 28 may. 2010

[18] Jena, available at http://jena.sourceforge.net/, accessed: 28 may. 2010

[19] HermiT reasoner, http://www.hermit-reasoner.com/, accessed: 28 may. 2010