Symptom Analysis of a Web Server Log

Jonice Oliveira[1], Wallace A. Pinheiro[1,3], Jano M. de Souza[1,2], Geraldo B. Xexéo[1,2], Marcelo Perazolo[4]

[1] Computer Science Department, Graduate School of Engineering
Federal University of Rio de Janeiro (UFRJ)
PO Box 68.511 - ZIP code: 21945-970 – Rio de Janeiro, RJ - Brazil

[2] Computer Science Department, Institute of Mathematics
Federal University of Rio de Janeiro – Brazil

[3] Computer Systems Engineering Section - SE/9
Military Institute of Engineering (IME)
Pr. General Tiburcio, 80 - ZIP code: 22290-270 - Rio de Janeiro, RJ – Brazil

[4] IBM Corporation – RTP, NC, USA

i@cos.ufrj.br

mperazol@us.ibm.com

Abstract

Problem identification is an area of research from Autonomic Computing. This work presents an approach based on Symptom Ontologies to facilitate problems identification and solution prediction. The proposal is to use symptom ontology on clickstream analysis, and provide resources for a site to have autonomic actions. To illustrate this idea, two differents symptoms from the clickstream analyses are described. Rules are used to recognize these symptoms, as well their possible effects are analyzed. These effects can correspond to either actions or recommendations that may correct a problem.

1 – INTRODUCTION

Clickstream analysis involves to collect and analyze clicks made by users of a site. This analyses can report aggregate data about which pages users visit in what order - which are the result of the succession of mouse clicks each user makes (that is, the clickstream).

Because a large volume of data can be gathered through clickstream analysis, many applications use datawarehouses to keep and organize these data. Normally, these data are store in Web Server logs. There are a lot of commercial applications that focus the Web log analysis, but most of them are quite limited when the focus is the recommendation or solution analysis.

IT professionals often seek ways to effectively detect and prevent problems associated to the applications and solutions they manage. New industry initiatives, like the Autonomic Computing paradigm, provide concepts such as Common Events and Symptoms to facilitate automatic processing, and consequently, identification, prediction and treatment of problems and incidents. But a big problem is the detection and description of effective symptoms. Some problems are easily inferred by human experience and symptoms can then be easily created, but in other cases the huge complexity of IT solutions nowadays makes it impracticable for domain experts to know beforehand what kinds of incidents, and consequently symptoms, they can expect from their environment. To solve this problem we can employ system verification methodologies and possible solutions' reasoning, based on past cases (like the Case-Based Reasoning paradigm).

This kind of solution, which is based on monitoring and reasoning, needs a common and well-defined knowledge representation of the domain. For that, we can use ontology, which is a specification of a conceptualization. That is, ontology is a formal description of the concepts and relationships that represent a domain. And for identification, prediction and treatment of problems and incidents in IT elements one requires a Symptoms Ontology, which consists of elements that describe what symptoms are and what their relationships are.

This work is a proposal of using Symptom Ontologies in a clickstream analysis to facilitate problem identification and solution prediction. For this purpose, some principles will be explained in section 2 and our proposal will be described in section 3. Opportunities and challenges, and the conclusion and future works are outlined in section 4.

2 – BIBLIOGRAPHIC REVIEW

This work is based on some principles, which will be explained in the following sub-sections.

2.1 – CLIKSTREAM AND WEB MINING

Etzioni (1996) emphasizes to use data mining techniques to automatically discover and extraction of information from documents and services into web. The information presents in the web can be exploited using differents ways. Nowadays, three different approaches are used: exploiting by content, exploiting by structure and exploiting by usage. In the exploiting by content are used techniques to aid users to find

documents in the Web according with predefined criteria. Texts, images, audios, videos, metadatas and hyperlinks are considered in this analysis. The exploiting by structure tries to discover structures of hyperlinks among documents, once that the Web consists not only of pages, but also of hyperlinks pointing from one page to another. These hyperlinks contain an enormous amount of latent human annotation. A hyperlink pointing to another Web page can be considered as the author's endorsement of the other page. Many search machines use this analyses to help in the building of their rank criterion. The third approach, the exploiting by usage, is also called Web log mining. It uses mining techniques to discover interesting usage patterns from users surf on the Internet. The clickstream analysis, a sub-area of the Web log mining, researches the interaction between Web users and sites. It considers that mouse clicks provide important indicatives about activities and events that happen during user visit. This paper will focus this analysis under the perspective of symptoms.

2.2– ONTOLOGIES

The advent of the Web has focused the attention of the computer scientist community towards the development of technologies that also enable its use by machines and not only focusing on humans. It required standards and special mechanisms to represent semantic information content on the Web. Albeit still a challenge research progresses towards the Semantic Web, such as that envisioned by Berners-Lee (2001).

Ontologies are central in the vision of the Semantic Web, since they enable knowledge sharing, reuse and common understanding between agents (human or machine), by providing a consensual and formal conceptualization of a given domain.

Many works have been done in the semantic web area, by creating standard languages and useful ontologies of real world concepts and systems. Some of the initiatives in this area come from the Dublin Core (Dublin Core Metadata Initiative, 2006), the Knowledge Interchange Format (KIF) (American National Standard, 2006), the Ontology Inference Layer (OIL) (W3C, 2001), the DARPA Agent Markup Language (DAML) (DARPA, 2006), the confluence of DAML and OIL that created the DAML+OIL reference (W3C, 2001), culminating in the Web Ontology Language recommendation (OWL).

The W3C approach to the semantic web is an evolutionary approach based upon the enhancement of existing languages so that each new layer builds upon the lower layer by adding additional semantic capabilities. The idea is to start with self-describing capabilities provided by existing assets like XML, Namespaces, URI, and XML Schema, then add data description capabilities with RDF and RDF Schema, and evolve into vocabulary building and associations provided by OWL. The model goes even further by describing a vision where future enhancements or extensions may provide logical and proofing capabilities and evolve all the way to provide trust and confidence in the machine's decisions.

The Semantic Web view of the W3C is built upon a series of fundamental principles, listed here:

1. Everything may be identified by URI's

2. Resources and links may have types

3. Partial information is tolerated

4. There is no need for absolute truth

5. Evolution is supported

Ontology artifacts help in the definition of relationships among elements. These artifacts are common for most ontology description languages, but, in this paper, we will refer to the artifacts that exist in the W3C set of recommended semantic web languages, namely XML, XML Schema, RDF, RDF Schema and OWL. These artifacts are:

Classes OWL classes are used to describe the concepts of domain ontologies, and are the starting point for most taxonomy collections. Classes are organized in hierarchies of sub-classes, and these hierarchies correspond to the taxonomical variation of the ontology. In the symptoms ontology, the OWL class hierarchy defines the taxonomy of any given symptom instance.

Individuals OWL individuals are used to describe specific instances that are leaves of the taxonomy tree. Individuals do not define specific properties and restrictions, but they inherit those of the class they represent. In the symptoms ontology, there are many individuals, and they are associated with the many leaf nodes in the symptoms taxonomy. Symptom individuals do not support specialization; in order to do that a class must be used.

Properties OWL properties are used to describe facts in the ontology. There are two kinds of properties: object properties and datatype properties. Object properties are used to describe relationships between two different resources. Datatype properties are used to describe relationships among a resource and an RDF or XML datatype (literal). In addition, datatype properties may be associated to individuals to designate their relationship with values of a certain datatype. In the symptoms ontology, properties are defined for each of the various classes component of the taxonomy.

Restrictions OWL restrictions come in many flavors. There are existential restrictions, universal restrictions, cardinality restrictions and value restrictions. The existential restriction characterizes elements where at least one value is in accordance with the relationship (*someValuesFrom*), where the universal restriction characterizes elements where all values are in accordance with the relationship (*allValuesFrom*). The cardinality restrictions are used to designate cardinality requirements in relationships, and value restrictions are used to designate that a specific value must be associated to a certain relationship.

When used wisely, restrictions may be a powerful tool in order to define particular semantics representation and reasoning. The focus of this research is to identify and predict problems in a grid environment using an ontology of symptoms.

2.3 – SYMPTOMS AND EVENTS

IT professionals often seek ways to effectively detect and prevent problems associated to the applications and solutions they manage. Novel industry initiatives, like the Autonomic Computing paradigm provide concepts such as Common Events and

Symptoms to facilitate automatic processing, and consequently, identification, prediction and treatment of problems and incidents. A big problem, however, is the detection and description of effective symptoms to start with. Some problems are easily inferred by human experience and symptoms can then be easily created, but in other cases the huge complexity of IT solutions nowadays makes it impracticable for domain experts to know beforehand what kinds of incidents, and consequently symptoms, they can expect in their environment. To solve this problem we can employ system verification methodologies and possible solutions' reasoning, based on past cases (like the Case-Based Reasoning paradigm).

This kind of solution, which is based on monitoring and reasoning, needs a common and well defined knowledge representation about the domain. For it we can use ontology, which is a specification of a conceptualization. That is, ontology is a formal description of the concepts and relationships that represent a domain. And for identification, prediction and treatment of problems and incidents in IT elements one needs a Symptoms Ontology.

Before we continue, it is important to know what the events and symptoms in the context of this paper mean. Event is an occurrence related to the functioning of a computing node, i.e., a peer in a grid. A symptom, which comes from an event, is the identification that a possible problem occurred. In the Symptom Analysis approach, it may also be created from state data, metrics or log records. For instance, when a user starts a program (event) and the computer that runs this program stops working (symptom) because of the program, we have examples of an event and a symptom caused by this event. Event is an action or occurrence to which a system can respond; examples include clicks, key presses and mouse movements. A symptom is a form of knowledge that indicates an incident. It could become a possible problem or situation in the managed environment. Basically, symptoms are indications of the existence of something else. They are often associated to an underlying autonomic computing architecture, where they are first order elements of the analysis part of an autonomic control loop. A comprehensive symptom reference model for autonomic computing can be found in reference Berners-Lee (2001) and Perazolo (2006).

In autonomic computing, symptoms are recognized in the monitor component of the control loop and used as a basis for an autonomic analysis of a problem or a goal. Symptoms are based on predefined elements (definitions and descriptions) provided to the autonomic manager, along with data that the monitoring infrastructure collects from managed resources, like events, as shown in Figure 1. The symptom definition expresses the conditions used by the monitor component to recognize the existence of a symptom, and the symptom description specifies the unique characteristics of a particular symptom that is recognized.

Most of the time, symptoms are closely connected to the self-healing discipline of autonomic computing because their primary intent is to indicate a problem, which is in the realm of the self-healing discipline. Symptoms can also be used, however, as triggers for other kinds of problems, such as those of other disciplines - self-protecting, self-optimizing, and self-configuring. Virtually all kinds of problems or predictions may start due to the occurrence of a symptom.

In autonomic computing architecture, classes of situations that can be handled in an autonomic way are often called self-*, meaning that different and varied IT

functions can be automated and managed by the component itself. The fundamental features, sometimes called self-CHOP (Miller, 2005) are: Self-configuring, Self-healing, Self-optimizing and Self-protecting.

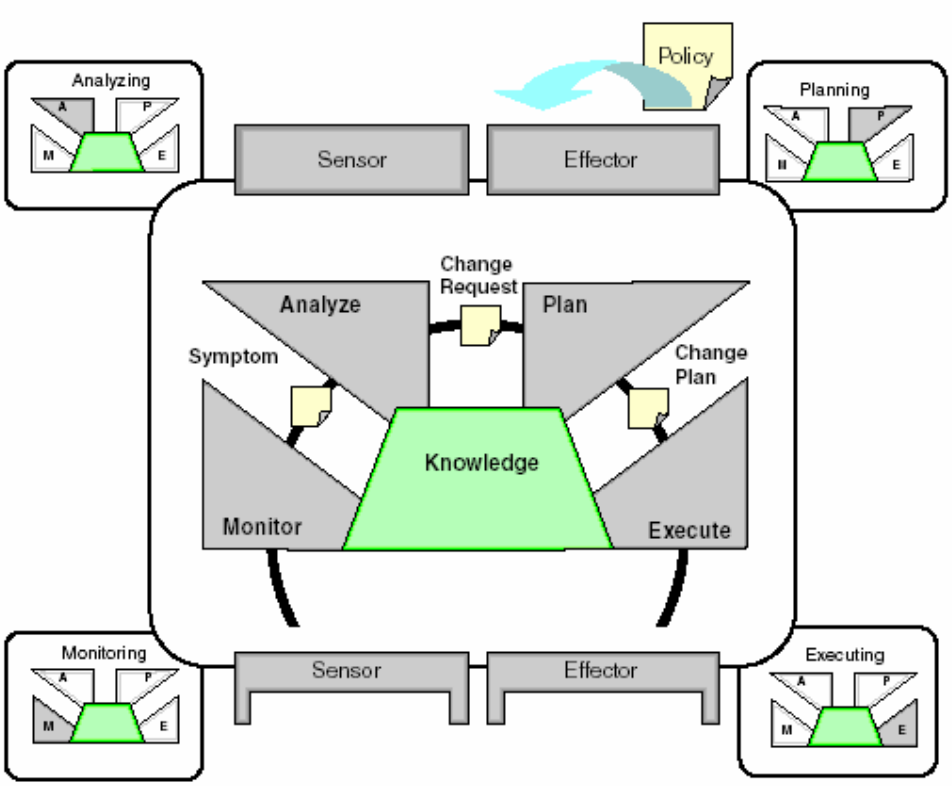

Figure 1 - Functional Details of Autonomic Manager (Perazolo, 2006)

Taxonomies and methods of classification of autonomic computing symptoms (or information that may be construed as similar to autonomic computing symptoms) exist today but they stop at the classification of individuals according to their respective categories. Examples of existing classification methods and taxonomies can be examined in American National Standard (2006) and Semanticweb (2006).

Figure 2 detaches an ontology obtained from an identification of symptom classes, properties and individual elements which can best describe the semantics associated to the information present in a symptom and expand these definitions.

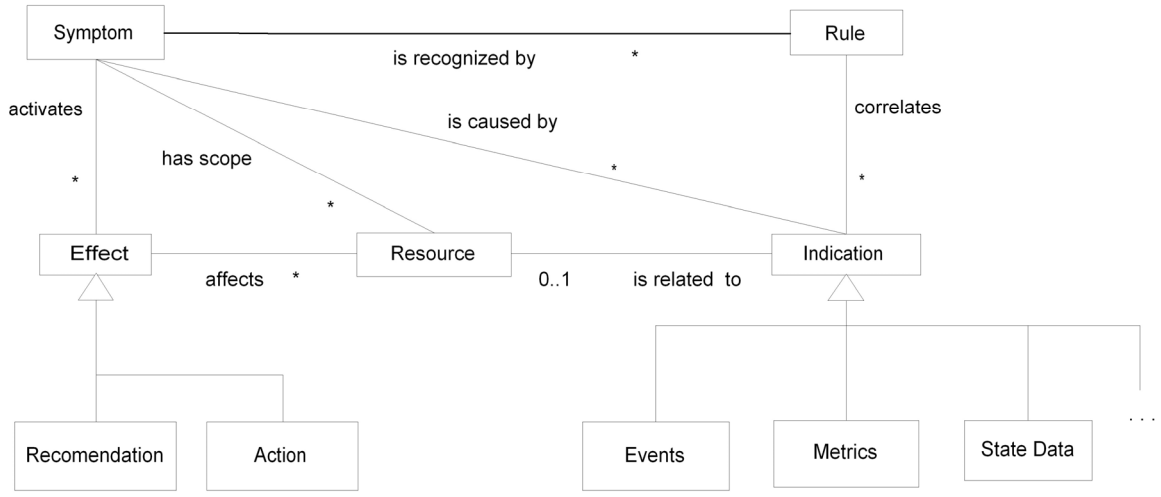

Figure 2. Base symptoms ontology

A concise explanation of each element into this ontology is made as follow:

- ***Symptom -*** It contains the information used to describe the symptom, along with information that characterizes the scope and importance attributed to the symptom.

- ***Rule -***This element contains the rules that are used in order to identify the situation or problem leading to the symptom. There are many existing rule formats in the industry or science realms that can be used as a form of recognition. The symptoms´ reference format is flexible enough to support multiple rule formats associated to a single symptom definition, so different runtime systems supporting different rules engines can work with their supported rule formats.

- ***Effect*** - This element contains the actions, tasks, processes, scripts or recommendations associated to the symptom. They are the effect that a symptom intends to cause in its surrounding system. It may be an active effect that will change the state of the system (run a program to solve a problem), or a passive one (display a message to a human operator), for information purposes or perhaps part of a tracking strategy. Symptom effects are executed once a symptom is recognized. The symptoms´ reference format is flexible enough to support multiple effects´ formats associated to a single symptom definition, so different runtime systems supporting different execution engines can work with their supported effect formats.

- ***Resource -*** Two different types of resources may exist. Unmanaged resources are those that are only conceptual and labeled in order to bring model organization. On the other hand, managed resources are part of a managed system and consequently have operations associated to them. When a managed resource has operations, they may be also construed as actions in the extended symptoms ontology and act as symptom effect elements.

- ***Indication -*** In this ontology we see that each symptom is caused by anonymous indications that can spawn their own ontology by themselves. Indications in this context are metrics, events or other form of data that can be acquired in the analysis context of a symptom. It may even be a SQL query in a database table for example.

It is interesting to notice that some questions about problems that are symptoms could

be answered with this ontology:

- **What** is this **symptom**? Here is the symptom itself, describing the incident.
- **Where** is the troublesome **resource**? All resources, especially in grid, need to be located and identified. This kind of question tries to show where the symptom happens;
- **When** is mapped by the **indication** that corresponds to an event, the temporal space;
- **Why** is related to the **rule** that identifies the situation or problem leading to the symptom;
- **Who** identifies anyone who has the competences needed to fix a problem, as a technician;
- **How** describes a set of events which results in the problem, how it is mapped by the **effect** that a symptom cause and the metrics associated.

3 – THE APPROACH

The proposal of this work is to use symptom ontology on clickstream analysis, and provide resources for a site to have autonomic actions, as described in section 2.3.

3.1 – CLICKSTREAM ANALYSIS

The main goal of the clickstream analysis is increase the quality of interactions with the user, what may cause more loyalty and profits. There are a lot of applications from this analysis like, for example: identify potential prime advertisement locations, improve the information service delivery, identify potential consumers in e-commerce sites, improve the sites' designers, anticipate users actions, improve the personalization of sites, and frauds and intrusions detections.

The content of log files can range according to the pattern chosen by the system administrator and Web servers can record all requisition made by users, therefore the amount of data can reach huge values. The main formats of logs available are: W3C Extended Log Format, NCSA Common Log File Format and Microsoft IIS Log Format. The next figure shows an example of a typical log server with the format:

<ip_addr><base_url>-<date><method><file><protocol><code><byte><referrer><user_agent>.

```
201.30.5.144 www.example.news.org - [11/Jun/2005:19:41:04 -0300] "GET / Calls/CDROM.html
HTTP/1.0" 200 799 'http://www.lycos.com/cgi-bin/pursuit?query=advertising+psychology:maxhits=20&cat=dir"
"Mozilla/4.5 [en] (win98, I)"

201.30.5.144 www.example.news.org - [11/Jun/2005:19:41:08 -0300] "GET /Calls/Images/figure1.gif
HTTP/1.0" 200 10689 'http://www.examples.news.org/Calls/CDROM.html"
"Mozilla/4.5 [en] (win98, I)"

201.30.5.144 www.example.news.org - [11/Jun/2005:19:42:12 -0300] "GET / HTTP/1.0" 200 4980 " "
"Mozilla/4.5 [en] (win98, I)"

201.30.5.144 www.example.news.org - [11/Jun/2005:19:41:08 -0300] "GET /Images/line1.gif
HTTP/1.0" 200 104 'http://www.examples.news.org/'
"Mozilla/4.5 [en] (win98, I)"
```

Figure 3. A Typical Server Log

Sweiger (2002) says that log files are dream data sources, once that are collected automatically for any Web server. The main advantages to Web server logs to behavior analysis are:

- They recreate the behavior of an real environment;
- The data are collected just observing behaviors, without any intervention to the users;
- The log files are free of disturbing insert by search engines;
- Information is store sequentially, obeying temporal patterns.

The actual tools of clickstream analysis offer ways to cleaning and transformation the fundamental data present in a log, for this they demand knowledge about the site structure and the application. Consequently, they also demand a human intervention to detect and solve similar problems every time. These applications do not permit to store and share strategies used early. They normally provide trivial analysis like, for example: time of visits, number of visited pages, pages more visited, amount of errors, metrics about performance and traffic, statistics about sessions.

The approach used in this work permits to detect potential problems and provides a way to store, share and recommend strategies used early to solve problems. This approach could use data directly extracted from either a Web server logs or applications that pre-processing logs.

3.2 - THE PROBLEM UNDER A SYMPTOMS-EVENTS VIEW

To give an example for the context describe above, imagine the following situation:

In a site of e-commerce, the main page shows the categories of products to sell. These categories obey a hierarchy. For example, a “video camera” is a subcategory of “photography” that is a subcategory of “equipment” that is a subcategory of “entertainment”. “Entertainment” is one of the main categories. This site is facing a problem, because nobody visits the page that presents the video cameras. A clickstream analyses provides this and other data interesting: pages relationed with the electronics main category are the most accessed in the site. The human analyst gathers all this data and decides to create a link to the category “video camera” from the category

"electronics". After this, the video camera starts to be sold in large numbers. But in the same category "electronics", a clickstream analysis says that a subcategory "VCR" is visited but not sold. A human analyst decides that is not worthwhile to continue selling VCRs, because nobody wants to buy them. Here, is possible to observe three differents elements: detection of the problem (it could be mapped like an indication in the base symptoms ontology), analysis of the possible solutions (it could be done by the rules in the base symptoms ontology) and the application of the appropriate solution (it corresponds to the element effect in the base symptoms ontology).

To emphasize the applicability of the base symptoms ontology in this environment, the "link broken" and "obsolete or too expensive" symptoms and will be described in details.

Some events could be related to these symptoms as: no hits to a page after one day (caused by a link broken), no sells to a page after one week (caused by either a obsolete or a expensive product). In this last case, it is possible use some metrics, such as "the percentage of sells to a product in a month" and "the average price of similar products" to indicate this symptom.

Rules could be used to recognize these symptoms as:

- "If a page that announces a product is not accessible then the link to this page is broken."; and
- "If a page that announces a product is accessible and the product does not receive an offer then a product announced in this page is obsolete or too expensive."

Following this idea, some effects could be foreseen in the ontology like recommendations (for example: stop to announce this product) or actions (for example: fix the link). The table 1 shows the relationship among the elements.

Table 1: Symptoms, Rules and Effects

SYMPTOMS	RULES	EFFECTS (ACTIONS OR RECOMMENDATIONS)
Link broken.	If a page that announces a product is not accessible then the link to this page is broken.	Fix the link.
Product obsolete or too expensive.	If a page that announces a product is accessible and the product does not receive an offer then a product announced in this page is obsolete or too expensive.	Stop to announce this product.

Finally, it is possible to map resources to the element resource in the ontology, like, for example, the computer responsible by host the web server and devices that make the communication environment.

This entire relationship between the base symptoms ontology and events sequence, and symptoms related with a global query in a grid environment is presented in Figure 4.

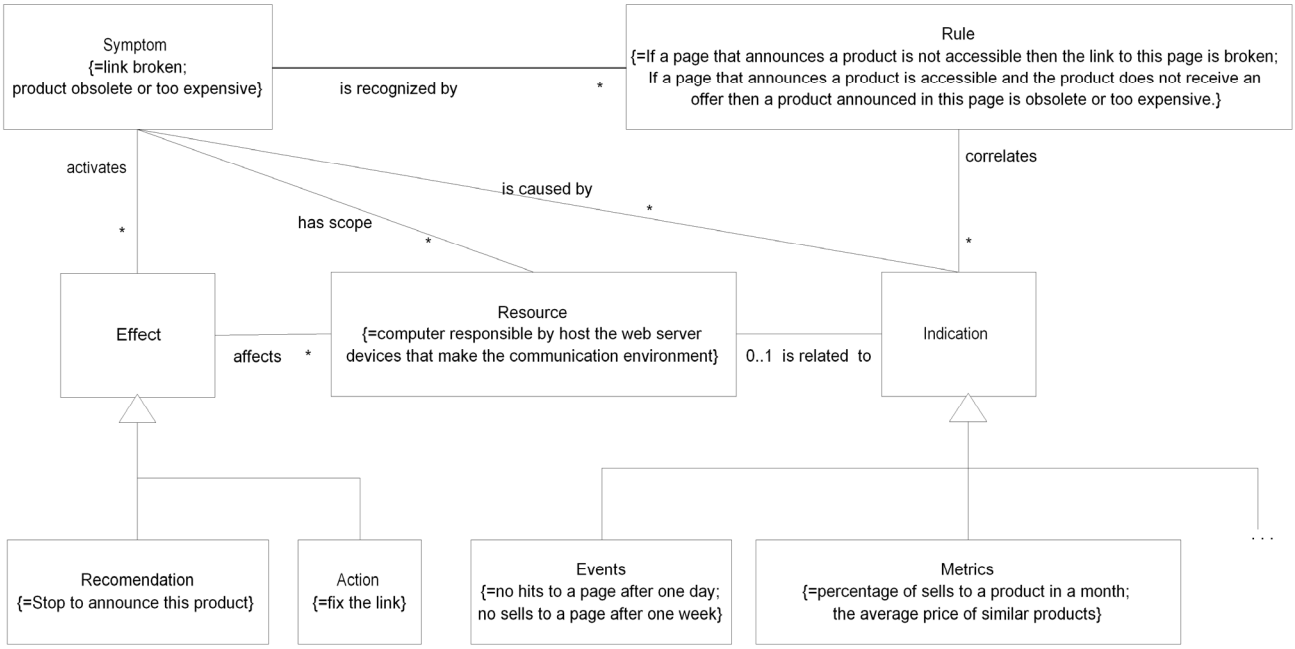

Figure 4. Diagram mapping between base symptoms ontology and events sequence and symptoms.

This is an ongoing work and several hurdles will be overcome in the future. One of them is exactly in the part of learning what a symptom is and translating it into a rule. Another issue is learning how humans solve the problems. In the present scenario, the mechanism for translating the knowledge is executed by a human agent, such as a technician.

In this context we can identify two distinct types of monitored information: raw events and correlated events. A raw event is merely an observation of a certain condition in a monitored resource and is associated with status and reporting conditions. In contrast, correlated events aim at including a certain level of intelligence to the data being reported. A correlated event is something that is, eventually, very similar to a symptom, but it is identified in lower layers that are near to the monitored resource itself.

We work with a 1-to-1 mapping between correlated events and specific symptoms associated to a particular resource type. Continuing in the scenario, these events enter the autonomic manager process, where they are transformed into symptoms in the "monitoring" block. These symptoms can then be further correlated together or with other events to form root-cause symptoms, or incidents.

4 – ACKNOWLEDGEMENTS

This research has the support of CAPES (The Brazilian Coordination for Postgraduate Staff Improvement), CNPq (The Brazilian Council for Scientific and Technological Development) and IBM through its Ph.D. Fellowship Program.

5 – CONCLUSION AND FUTURE WORK

The base of Symptoms Ontology consists of elements that describe what symptoms are and what their relationships are. The purpose of this work is to determine whether the essence of IT problem determination and prediction can be captured through a limited number of generic symptom types. Symptoms with nearly identical structures, such as "application connection error" and "server unreachable error" are to be classified as generic types, e.g. "network condition X". In this study we analyze a kind of solution deployed by Web server logs.

Some of our future work comprises the learning process of symptom identification and problem solution. It means learning what a symptom is and translating it into rules. Another issue is learning how humans solve the problems. In the present scenario, the mechanism for translating the knowledge of the technician (identifying and solving the problem) is reported by a human agent, such as a technician.

References

American National Standard, "The Knowledge Interchange Format draft proposed," 2006.

T.Berners-Lee, J.Hendler, and O.Lassila, "The Semantic Web," Scientific American: 2001.

DARPA, "The DARPA Agent Markup Language," in DARPA's Information Exploitation Office, ed, 2006.

Dublin Core Metadata Initiative, "Documents," 2006.

O.Etzioni, "The World Wide Web: quagmire or gold mine?," 1996.

B.Miller, "The Autonomic computing edge: Can you CHOP up autonomic computing," http://www-128.ibm.com/developerworks/autonomic/library/ac-edge4, 2005.

M.Perazolo, "Symptoms deep dive, Part 1: The autonomic computing symptoms format," http://www-128.ibm.com/developerworks/autonomic/library/ac-symptom1/: 2006.

Semanticweb, "The Ontology Inference Layer Initiative," 2006.

M.Sweiger and et al, Clickstream Data Warehousing, Wiley Computer Publishing: New York, 2002.

W3C, "The DAML+OIL Reference Description," 2001.