Specification of a MIB XML for Systems Management

Andrey Soares, MSc.
Federal University of Santa Catarina – UFSC
andrey@inf.ufsc.br

Marcello Thiry, Dr.
University of the Valley of Itajai – UNIVALI
thiry@sj.univali.br

Abstract

This work presents an XML application for systems management. The main purpose is showing how to map the information from a MIB to a XML document. It introduces system management architecture based on SNMP and XML agents. According to the MIB proposed a DTD file, containing the available information structure, is specified to represent the management information. The mapping from MIB to XML document follows some rules: identifiers are defined to mapping the management information representation; the script file is created to associate the information from the DTD file to an identifier; The collect file is created to represent the MIB with the information collected automatically from the system management; the XML file is created from the DTD file using the script (contents) and collect (information) files. A web application is also presented to show the use of XML document as a MIB and its results.

Keywords: XML, MIB, Systems Management

1. Introduction

The growth of the computer network use has been generating a demand for the management software use of nets and systems management. However, some factors has been contributed for the search of another solution to systems management, like the difficulty to find a MIB (Management Information Base) which covers all the needs of a company, the dependence of the standard SNMP (Simple Network Management Protocol), etc.

XML (Extended Markup Language) can be a solution for system management by introducing the characteristic of being multi-platform and with it's use of a heterogeneous systems.

The utilization of the XML document should enable more flexibility in the utilization of existing MIB, as well as in the creation and adaptation of the information.

The main goal of this project is to elaborate and to introduce XML's utilization as an information storage and manipulation tool, in other words, a MIB-XML.

2. Architecture

Figure 1 presents an outline of the utilization of XML and SNMP that can be used to manipulate information of a MIB; this MIB could be Standardized (MIB S) or Not Standardized (MIB NS).

According to WEBBER [11], and it's understanding of a Standardized MIB, that MIB must fall within these guidelines:

- Which was written correctly;
- Which has been verified regarding the syntax and semantic through a compiler;
- Which introduces conformity with SMI (Structure of Management Information);
- Which can be accessed through primitive SNMP.

Webber's definition for a Not-Standardized MIB is, the MIB must be created by a company with the objective of collecting information for a specific purpose without preoccupation with standardization. For example, a program written by an employee to collect information of a management object that is capable of recording this information in a text file.

This work introduces four possibilities for the management of a MIB:

1. **SNMP Requisition**: The traditional method of management, where management software manipulates information from a Standardized MIB, through SNMP primitives.
2. **SNMP-XML Mapping**: Management software that wants to manipulate information of a MIB Not-Standardized; however the management software only uses SNMP primitives. In this case, the SNMP agent receives the SNMP requisitions coming from the management software and

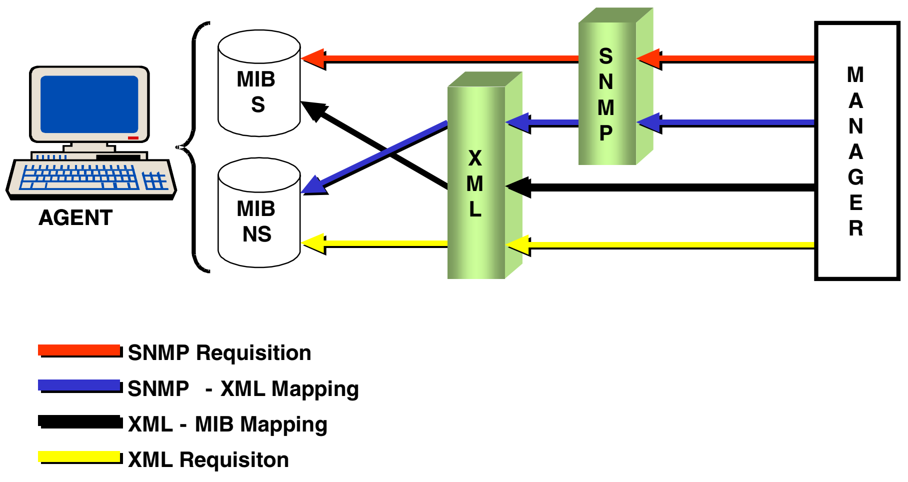


Figure 1: XML - SNMP Mapping outline

then forwards them to the XML agent. Once the information is obtained, the XML agent returns the information to the SNMP agent which return answers the solicitation of the management software.

3. **XML-MIB Mapping**: Standardized MIB collecting information of equipment, but not having management software that does SNMP requisitions. In this case, XML requisitions for the XML agent are done. The XML agent converts the requisitions in primitive SNMP and does the requisitions for the MIB.
4. **XML Requisition**: A Not-Standardized MIB and no management software. In this case, XML requisitions for the XML agent are done. This then seeks the information and returns it to the applicant.

3. Proposed MIB

According to MILLER [9], a MIB is a structure that contains the necessary variables to monitor or to manage the components in a net. In this case, it was an elaborated analyzation of the available information in a workstation using Linux operating system that can identify and define the information that is important for the management of the station.

Some information was identified as belonging to the systems management. However, it was chosen as a subset of this information to allow the validation of this model. The choice was made with base in the tests environment used for the project. For example, it is considered that each computer used will allow the use of one processor only, but it is possible for the existence of more than one network adapter

For PEITER [10], the existing MIB for nets and systems management, do not offer the flexibility to attend all the needs to different systems. It is difficult to get a management application that solves the specific problems of each company. Therefore, many companies create the own MIB, customized with the required information. The created MIB proceeds being a Not-Standardized MIB due to the difficulty and the time spent and the compilation and implementation of a Standardized MIB.

The objects used to be managed and stored in a MIB were divided into 4 categories:

1- COMPUTER: The information concerning the computer. The attributes are:

Hostname: name that identifies the computer in the net

Kernel: version of the kernel

2- CPU: The information that identifies the processor used in the computer and its utilization. The attributes are:

Processor: kind of processor (Example: Pentium)

Speed: speed in MHz of the processor

Manufacturer: name of the manufacturer of the processor (Example: Intel)

OccupiedUsr: percentile of CPU (Central Process Unit) occupied by the user

OccupiedSys: percentile of CPU used by the operating system

Lazy: percentile of free CPU for use

3- MEMORY: The information that shows the utilization of the memories: RAM (Random Access Memory) and SWAP. The attributes are:

Total: quantity of total RAM memory (in Kbytes)

Used: quantity of busy RAM memory (in Kbytes)

Liberate: quantity of free RAM memory (in Kbytes)

SwapTotal: memory quantity SWAP total (in Kbytes)

INFORMATION	DTD
	<!ELEMENT **MIB** (COMPUTER)>
COMPUTER	<!ELEMENT **COMPUTER** (HOSTNAME, KERNEL, CPU, MEMORY, NETWORKADAPTER)>
HostName	<!ELEMENT **HOSTNAME** (#PCDATA)>
Kernel	<!ELEMENT **KERNEL** (#PCDATA)>
CPU	<!ELEMENT **CPU** (PROCESSOR, SPEED, MANUFACTURER, OCCUPIEDUSR, OCCUPIEDSYS, LAZY)>
Processor	<!ELEMENT **PROCESSOR** (#PCDATA)>
Speed	<!ELEMENT **SPEED** (#PCDATA)>
Manufacturer	<!ELEMENT **MANUFACTURER** (#PCDATA)>
Occupied Usr	<!ELEMENT **OCCUPIEDUSR** (#PCDATA)>
Occupied Sys	<!ELEMENT **OCCUPIEDSYS** (#PCDATA)>
Lazy	<!ELEMENT **LAZY** (#PCDATA)>
MEMORY	<!ELEMENT **MEMORY** (TOTAL, USED, LIBERATE, SWAPTOTAL, SWAPFREE)>
Total	<!ELEMENT **TOTAL** (#PCDATA)>
Used	<!ELEMENT **USED** (#PCDATA)>
Liberate	<!ELEMENT **LIBERATE** (#PCDATA)>
SwapTotal	<!ELEMENT **SWAPTOTAL** (#PCDATA)>
SwapFree	<!ELEMENT **SWAPFREE** (#PCDATA)>
NETWORK	<!ELEMENT **NETWORK** (NETPLATE)*>
	<!ELEMENT **NETPLATE** (DEVICE, RECEIVED, TRANSMITTED, IP, IRQ, IO)>
Device	<!ELEMENT **DEVICE** (#PCDATA)>
Received	<!ELEMENT **RECEIVED** (#PCDATA)>
Transmitted	<!ELEMENT **TRANSMITTED** (#PCDATA)>
IP	<!ELEMENT **IP** (#PCDATA)>
IRQ	<!ELEMENT **IRQ** (#PCDATA)>
IO	<!ELEMENT **IO** (#PCDATA)>

Table 1: Creation of the DTD file

SwapFree: memory quantity SWAP free (in Kbytes)

4- NETWORK: The information that identifies the net interfaces available in the computer. The attributes are:

Device: name of the device that identifies the net interface (Example: eth0)

Received: packages quantity received by the net interface

Transmitted: packages quantity transmitted by the net interface

IP: IP (Internet Protocol) Address used by the net interface

IRQ: number of the interruption used by the net interface

IO: input and output address used by the net interface

The information selected can be modeled through a guided method for objects. For this project it was the adopted method defined in [3], which uses the UML (Unified Modeling Language) notation [4] [5].

4. XML

XML is a content language, defined in 1996 by W3C (World Wide Web Consortium), derivative of the SGML (Standard Generalized Markup Language), an international standard ISO/IEC 8879, for the structures and contents definition of electronic documents [6] [3] [1]. In February 1998, W3C launched the specification 1.0 of XML [6].

XML was initially considered a substitute of the HTML (Hypertext Markup Language), because this introduced several referring limitations to new elements creation, reutilization of documents and in the **content-information-presentation** relation.

According to LEIVA ET. Al. [8], a XML document is composite for:

- **CONTENT**: The file DTD (Document Type Definition) that owns the structure of the information, specifying the correct syntax of the document.
- **INFORMATION**: The XML file that owns the information structured as DTD.
- **PRESENTATION**: The XSL (Extensible Style sheet Language) and CSS (Cascading Style Sheet) files, used to define, as the information that will be introduced.

4.1. DTD file

The development of this project was based on specification 1.0 of XML that includes DTD as data definition language. It used the DTD because of its simplicity but it could have used XML Schema that supplies some DTD's Limitations, such as, the data kinds use as Integer, Float, Boolean, Date, etc. Besides introducing some parameters that allow the establishment of more safety to the information.

The DTD file is the specification of the structure and the syntax of a XML document [6]. According to LEIVA ET.AL. [8], a DTD contains basically elements and attributes. Where the declaration of an element defines the name, the composition of the sons' elements, contents and optionally the attributes [3].

The utilization of the information collected in a XML document initiates with the interpretation of the DTD file. In this file, for each information or information grouping, is a defining element. Thus, the information is specified by **<! ELEMENT attribute name (#PCDATA)>** and an information grouping is specified by **<! ELEMENT group name (atribute1, atribute2, …) >**.

In the Table 1, the first declaration in the DTD file is the most generic grouping, called MIB. This declaration can be interpreted as: **A MIB is composed of COMPUTER information.** Soon after, it is declared the COMPUTER attribute; it then made up the information HOSTNAME, KERNEL, CPU, MEMORY and NETWORK. For each COMPUTER attribute a specific declaration is created. Doing this, the next declaration is the attribute HOSTNAME. Since it is not a grouping, the element was defined as a #PCDATA (a string contend pure information). This procedure continues until all the information and groupings have been declared.

The chosen information contemplates more than one network adapter in the computer. This way, it was necessary to do a special treatment in the declaration of the NETWORK element. This element received a multiplicity symbol (*) at the end of its declaration. This can be interpreted as **The grouping NETWORK that can contain several network adapters**. Therefore, a new grouping was created and denominated NETPLATE,

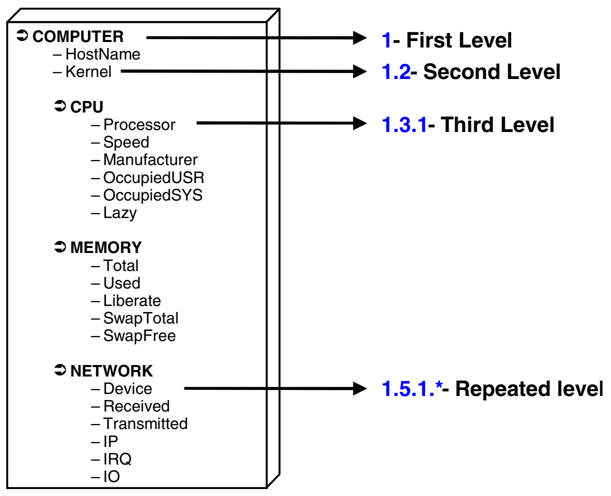


Figure 2: Identifiers Attribution

which contains the specific information of each network adapter installed in the computer.

The NETPLATE grouping then will be declared with

1- **COMPUTER**
  1.1- HostName
  1.2- Kernel

  1.3- **CPU**
    1.3.1- Processor
    1.3.2- Speed
    1.3.3- Manufacturer
    1.3.4- OccupiedUsr
    1.3.5- OccupiedSys
    1.3.6- Lazy

  1.4- **MEMORY**
    1.4.1- Total
    1.4.2- Used
    1.4.3- Liberate
    1.4.4- SwapTotal
    1.4.5- SwapFree

  1.5- **NETWORK**
    1.5.1.*- Device
    1.5.2.*- Received
    1.5.3.*- Transmitted
    1.5.4.*- IP
    1.5.5.*- IRQ
    1.5.6.*- IO

Figure 3: Identifiers structure in Not-Standardized MIB

the attributes: DEVICE, RECEIVED, TRANSMITTED, IP, IRQ and IO, which are declared as being #PCDATA.

From the classes' diagram, it is possible to arrive to the specification of the file DTD [3].

4.2. XML file

Once defined the structure of a XML document, through the DTD file, the next step is the creation of a file with the information in the XML format. This information is collected by the agents created for this task:

- From Standardized MIB, where will be necessary the requisitions utilization SNMP;
- From Not-Standardized MIB, where the information are extracted from a kind of text file.

With the definition of the information described in the DTD file and with the information stored in the MIB. The XML file will be generated automatically.

4.3. Identifiers Attribution

Initially, it becomes necessary to standardize the information. Thus, all information or information groups will receive an identifier. In case it is a Standardized MIB, it should use the file ASN.1 (Abstract Syntax Notation One) relative MIB's Definition and to follow the same identifiers structure OID (Object Identifier). In case it is a Not-Standardized MIB, it should specify the identifiers, as the structure created in the Figure 2.

In both cases, the information will be divided into levels represented by whole values initiated in 1. Each sub-level will be separated by dot (.) and it will be restarted the counting in 1. The information that repeat will be represented by asterisk (*), symbolizing the several occurrences existence. Considering the created MIB, the identifier structure is introduced in the Figure 3.

4.4. Script and Collect files

After defining the identifiers structure, the next step is to create the SCRIPT file. This file will match each element #PCDATA from the DTD file to an identifier. In the utilization of a Not-Standardized MIB, the COLLECT file also should be created in the utilization of the established identifiers. And in the utilization of a Standardized MIB, the information will be extracted directly from the MIB through SNMP requisitions. Table 2 introduces the content of the SCRIPT and COLLECT files.

SCRIPT	
ID	**Element**
1.1	HOSTNAME
1.2	KERNEL
1.3.1	PROCESOR
1.3.2	SPEED
1.3.3	MANUFACTURER
1.3.4	OCCUPIEDUSR
1.3.5	OCCUPIEDSYS
1.3.6	LAZY
1.4.1	TOTAL
1.4.2	USED
1.4.3	LIBERATE
1.4.4	SWAPTOTAL
1.4.5	SWAPFREE
1.5.1.*	DEVICE
1.5.2.*	RECEIVED
1.5.3.*	TRANSMITTED
1.5.4.*	IP
1.5.5.*	IRQ
1.5.6.*	IO

COLLECT	
Identifier	**Information**
1.1	Cerebro.andrey.com.br
1.2	2.40
1.3.1	Pentium 75 - 200
1.3.2	100.228
1.3.3	GenuineIntel
1.3.4	2.9%
1.3.5	1.9%
1.3.6	95.9%
1.4.1	65116K
1.4.2	61576K
1.4.3	3540K
1.4.4	104384K
1.4.5	104384K
1.5.1.1	eth0
1.5.2.1	1661
1.5.3.1	11
1.5.4.1	198.168.0.184
1.5.5.1	11
1.5.6.1	0x6100
1.5.1.2	eth1
1.5.2.2	0
1.5.3.2	0
1.5.4.2	198.168.0.2
1.5.5.2	9
1.5.6.2	0x6200

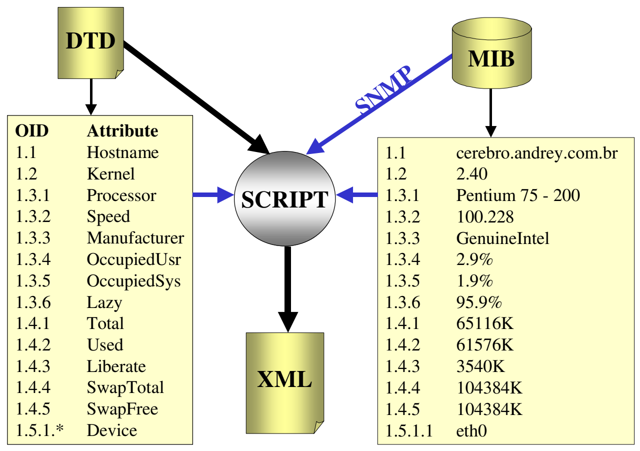
Table 2: Content of the SCRIPT and COLLECT files

The information made available in the COLLECT file should be added to the identifier. Doing this makes it possible for the relationship of the information of the DTD element with the collected information.

4.5. XML-MIB mapping

With the SCRIPT and COLLECT files properly established, the next step is the generation of the XML file from the structure of the DTD file. The SCRIPT and COLLECT files are used according to the relationship between content and information, as it is shown in Figure 4.

Basically, for each element declared in the DTD file a entrance will be created in the XML file. When the element is a #PCDATA, it will be activating a function that will seek the value of the information

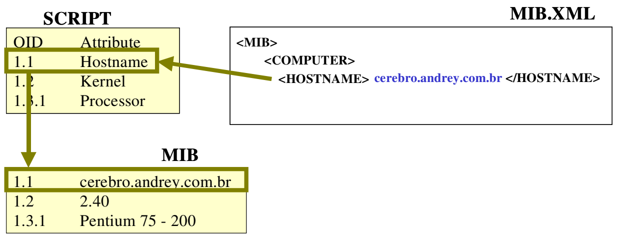


OID	Attribute
1.1	Hostname
1.2	Kernel
1.3.1	Processor
1.3.2	Speed
1.3.3	Manufacturer
1.3.4	OccupiedUsr
1.3.5	OccupiedSys
1.3.6	Lazy
1.4.1	Total
1.4.2	Used
1.4.3	Liberate
1.4.4	SwapTotal
1.4.5	SwapFree
1.5.1.*	Device

1.1	cerebro.andrey.com.br
1.2	2.40
1.3.1	Pentium 75 - 200
1.3.2	100.228
1.3.3	GenuineIntel
1.3.4	2.9%
1.3.5	1.9%
1.3.6	95.9%
1.4.1	65116K
1.4.2	61576K
1.4.3	3540K
1.4.4	104384K
1.4.5	104384K
1.5.1.1	eth0

Figure 4: XML – MIB Mapping

in MIB, where the function will search for the name of the element in the SCRIPT file, and this will inform which identifier is associate to the element. With the value of the identifier, the function then seeks the value of the element in the COLLECT file or in MIB SNMP through the GET primitive (Figure 5).



OID	Attribute
1.1	Hostname
1.2	Kernel
1.3.1	Processor

```
<MIB>
    <COMPUTER>
      <HOSTNAME> cerebro.andrey.com.br </HOSTNAME>
```

1.1	cerebro.andrey.com.br
1.2	2.40
1.3.1	Pentium 75 - 200

Figure 5: SCRIPT - MIB Relationship

The previous Figure introduces the content of the XML file. After the file is created, it is possible to do a validation of its content, through the consistency verification between the DTD and XML files.

5. Application

For the verification and validation of the model of proposed management, it will use the tests composite environment for two microcomputers interlinked in TCP/IP local net.

It initially tried Not-Standardized MIB management, in other words the MIB was created to manipulate and to store information, however without standardization.

One of the computers was designated to be an XML agent, acting as a server. Therefore, the HTTP service was configured in Linux environment. This service enables other computer access through the use of this protocol using a navigation browser through the Internet, such as

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE MIB SYSTEM "mib.dtd">
<MIB>
        <COMPUTER>
                <HOSTNAME>cerebro.andrey.com.br</HOSTNAME>
                <KERNEL>2.40</KERNEL>
                <CPU>
                        <PROCESSOR>Pentium 75 - 200</PROCESSOR>
                        <SPEED>100.228</SPEED>
                        <MANUFACTURER>GenuineIntel</MANUFACTURER>
                        <OCCUPIEDUSR>2.9%</OCCUPIEDUSR>
                        <OCCUPIEDSYS>1.9%</OCCUPIEDSYS>
                        <LAZY>95.9%</LAZY>
                </CPU>
                <MEMORY>
                        <TOTAL>65116K</TOTAL>
                        <USED>61576K</USED>
                        <LIBERATE>3540K</LIBERATE>
                        <SWAPTOTAL>104384K</SWAPTOTAL>
                        <SWAPFREE>104384K</SWAPFREE>
                </MEMORY>
                <NETWORK>
                        <NETPLATE>
                                <DEVICE>eth0</DEVICE>
                                <RECEIVED>1661</RECEIVED>
                                <TRANSMITTED>11</TRANSMITTED>
                                <IP>198.168.0.184</IP>
                                <IRQ>11</IRQ>
                                <IO>0X6100</IO>
                        </NETPLATE>
                        <NETPLATE>
                                <DEVICE>eth1</DEVICE>
                                <RECEIVED>0</RECEIVED>
                                <TRANSMITTED>0</TRANSMITTED>
                                <IP>198.168.0.2</IP>
                                <IRQ>9</IRQ>
                                <IO>0X6200</IO>
                        </NETPLATE>
                </NETWORK>
        </COMPUTER>
</MIB>
```

Figure 6: Content of the MIB.XML file

Netscape and Internet Explorer. This computer still remained an information collection agent; from time to time they collected and made available the information in the COLLECT file. Once the MIB.XML file is created, it is necessary to have an application to manipulate this file. It used the Internet browser Explorer 5.5 for utilization of the application and JavaScript language for the creation of functions that manipulates the XML file. It also could have used Netscape browser 6.0, where it would be necessary to verify which methods and properties of the DOM (Document Object Model) object are available for this browser. Following will be the introduction of available options in the created application:

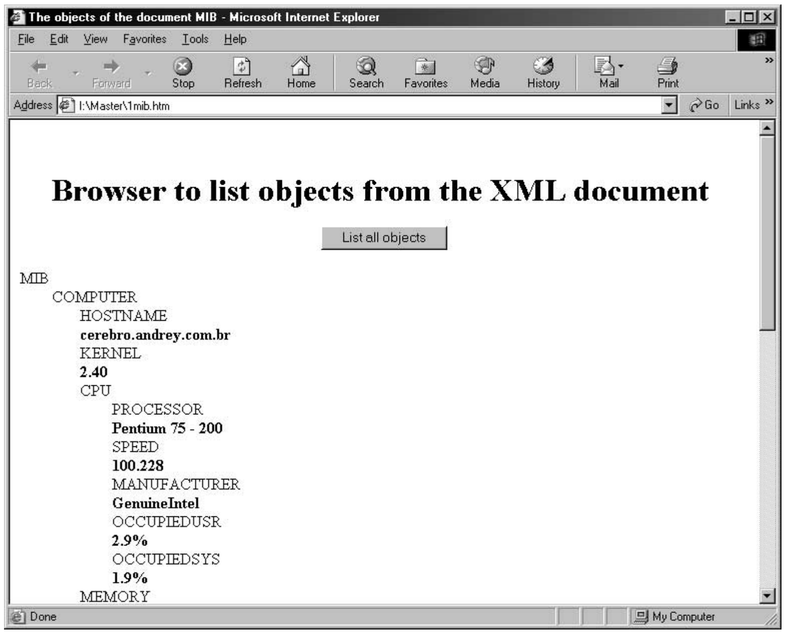
Visualize MIB.DTD: Introduces the content of the file MIB.DTD, which is used to represent the structure of the information that will be made available, by the MIB.XML file.

Visualize MIB.XML: Introduces the content of the MIB.XML file, which contains the information that will be made available.

Validation of the Document XML: Used to verify and to validate the consistency of the information between MIB.DTD and MIB.XML files. The documents validation consist in checking the object **parse Error** and it's properties. In case some inconsistency occurs a mistake message will be introduced indicating:

- The XML file that is being analyzed;
- The line where the mistake was found;
- The position of the cursor inside the line where the mistake was found;
- A message informing the reason why the mistake happened;
- The code of the mistake occurred.

Browser of the Document MIB.XML: Introduces the value of the variables of MIB, acting like a MIB Browser (Figure 7). The function used to list the elements


The objects of the document MIB - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print
Address I:\Master\1mib.htm
Go Links
Done
My Computer

Browser to list objects from the XML document

List all objects

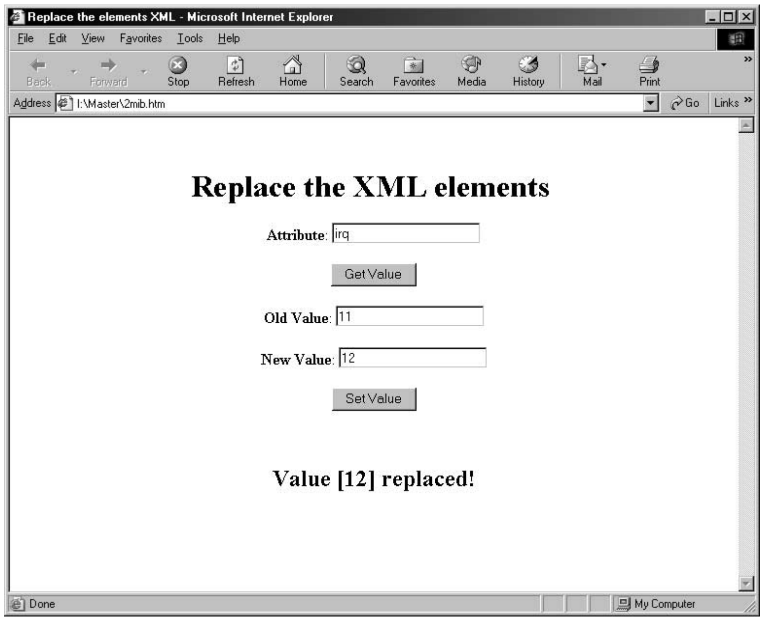
```
MIB
    COMPUTER
        HOSTNAME
        cerebro.andrey.com.br
        KERNEL
        2.40
        CPU
            PROCESSOR
            Pentium 75 - 200
            SPEED
            100.228
            MANUFACTURER
            GenuineIntel
            OCCUPIEDUSR
            2.9%
            OCCUPIEDSYS
            1.9%
        MEMORY
```

Figure 8: It shows the objects of the MIB.XML document

of the XML document, uses recursive mechanism to access all the elements of the document.

List Objects of the Document MIB.XML: according to WEBBER [11], the nets manager does not want to call a MIB Browser to start to inspect the variables. It would be more practical, if the manager ask information of a specific variable, because it would be necessary to navigate by the variables to arrive to the required element. To elaboration of this function, it will use the method **getElementByTag**, which return the object list of the past element as an argument. The argument is then displayed to the user.

Modify objects of the document MIB.XML: some variable of MIB can be used as control and allowed to


Replace the elements XML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print
Address I:\Master\2mib.htm
Go Links
Get Value
Set Value
Done
My Computer

Replace the XML elements

Attribute: irq

Old Value: 11

New Value: 12

Value [12] replaced!

Figure 7: Modifying the value of an element XML

be changed (Figure 8). Thus the user could, for example, close an application; incapacitate the door of a Switch, etc. The function, which modifies the value of an element uses the method **replace Data** to modify the value of the document, and the method **save** to record the document with the new changes.

6. Conclusions

This work searched a solution based on XML for a system open and multi-platform that allowed the management of heterogeneous systems. For sure, XML introduced great potential for system management; being highlighted the transparency between consultation and the implementation of the information made available. In this situation, the user or application does not need to know how the information is stored internally.

The introduced solution offers compatibility with the management systems already existing through automated tools, a MIB already defined and Standardized was quickly converted to XML documents with the minimum effort.

Still with idea of optimization, it was proposed that MIB's Mapping for XML follow some rules. Once converted MIB, it was elaborated a Web application to demonstrate the files for utilization and manipulation of XML. Thus, it was possible to verify the advantages in the utilization of the HTTP for information transfer, mostly with regard to shared information.

With the accomplished test, the management architecture based on XML showed apt as much as final tool as support tools in the management activities.

Several improvements and increases can be seen from the obtained results in this project:

- It verified that the files utilization DTD for the definition of the structure of the information owns some deficiencies, which can be solved with the use of XML Schema.
- They were elaborated for the XML document, the equivalent operations to GET and SET, staying as the next step for the elaboration of the TRAP operation.
- The pages introduced in the application are experimental, owing itself in a next stage, give emphasis to the visual aspects as the information is introduced. It suggests XSL and CSS.
- Besides the management addressed to the nets and boarded operating systems in this project; it can expand the utilization of this project, for example, the database management. Database management is where the user would do the mapping of the information that it wants to manage, using script to generation of XML files proposed in this work.

7. References

[1] BOSAK, John. XML, Java and the future of the Web. Sun Microsystems. March, 1997 URL: http://www.iblibio.org/pub/sun_info/stabdards/xml/wity/xmlapps.html

[2] BRAY, Tim. SPERBERG-MCQUEEN, C.M. Extensible Markup Language (XML). W3C *Working Draft*, November, 1996. URL: http://www.w3.org/tr/wd-xml-961114.html

[3] CARLSON, David. Modeling XML applications with UML: Pratical e-Business Applications. Addison-Wesley. 2001.

[4] ERIKSSON, Hans-Erik. PENKER, Magnus. UML ToolKit. Wiley. 1998.

[5] FOWLER, M. SCOTT, K. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Object Technology Series, 1999.

[6] HOLZNER, Steven. Inside XML. New Riders, 2001.

[7] JITAO, Hou. GDMO - Guidelilnes for Definition of Managed Objects. July, 2001. URL: http://www.whatis.techtarget.com/definition

[8] LEIVA, Willie. VICENTE, William. Aprendendo XML por meio de um Estudo de Caso. UNICAMP. Campinas, 1999. URL: http://www.dca.fee.unicamp.br/projects/sapiens/seminars/atas/register/sem9911/index.htm

[9] MILLER, Mark. Managing Internetworks with SNMP. IETF Network Management Documents. M&T Books, 1999.

[10] PEITER, Rui C. Um modelo de um agente SNMP para gerenciamento de rede. TCC- Univali-Ciência da Computação, 2000.

[11] WEBBER, Celso K. Uma Mib para aplicações Internet. Dissertação de Mestrado. CPGCC-UFSC, 1997