

# Efficiency Support for SMI-S with XML Database

*Ze Deng <sup>1)</sup>, Zhan Shi <sup>1)</sup>, Dan Feng <sup>1)</sup>, Hao Huang <sup>1)</sup>*

## Abstract

The Storage Management Initiative Specification (SMI-S) has been proposed for years to standardize the management of storage resources in Storage Area Network (SAN). In the SMI-S architecture, there are three main components: Service Agent (SA), Directory Agent (DA) and User Agent (UA). SA represents storage resource. DA collects storage service information from SAs and provides storage resource discovery service for UAs. UA monitors and manages related storage resources for applications. In order to guarantee DA efficiently provides storage resource discovery service for UAs and UA can efficiently manage discovered storage resources, in this paper, we propose two effective schemes to improve SMI-S management architecture based on XML database and related techniques.

## 1. Introduction

The SNIA Storage Management Initiative specification (SMI-S) provides for heterogeneous, functionally rich, reliable, and secure monitoring/control of mission critical global resources in complex and potentially broadly distributed multi-vendor Storage Area Network (SAN) topologies. As such, this interface overcomes the deficiencies associated with legacy management [1]. So, SMI-S is ideal for emerging ubiquitous storage resource management. The management architecture is shown in Figure 1.

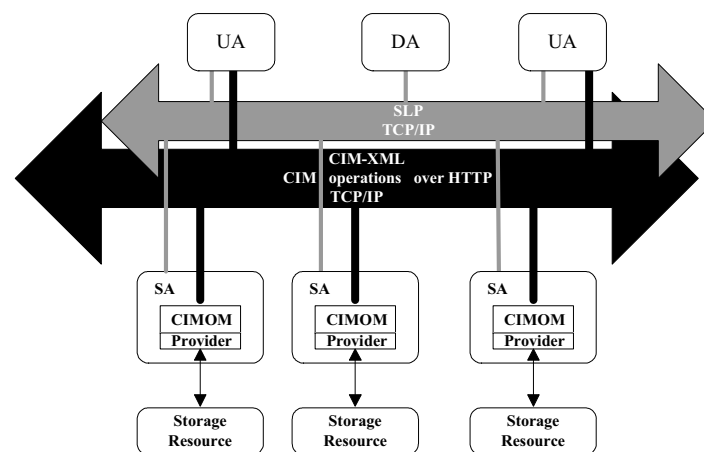


Figure 1: SMI-S Management Architecture

<sup>1</sup> Key Laboratory of Data Storage System, Ministry of Education, School of Computer, Huazhong University of Science and Technology, 1037 Luo Yu Road, Wuhan, Hubei, China 430074, deng\_ze@163.com, {zshi | dfeng}@hust.edu.cn

The SMI-S architecture mainly consists of three components: Service Agent (SA), Directory Agent (DA) and User Agent (UA). SA can describe storage resource by Common Information Model (CIM) [2], these storage resource CIM objects are stored into Common Information Model Object Manager (CIMOM) [3] resided in SA. Meanwhile, using Service location protocol (SLP) [4], SA advertises DA about its basic information including available service and location. DA is the one that collects the service information from SAs and sets up storage resources discovery service through SLP. So that UA can acquire and manage storage resources for user application. To manage required resources, UA firstly need query DA to obtain the locations of resources represented by SAs, and according to acquired locations, UA connects every SA to manage corresponding resource by CIM Operations over HTTP [5][6]. It is noted that for UAs, the management operations on resources are in fact on CIM objects. When CIM objects are managed, the corresponding resources can get management operation indications by Provider which is a specific communication interface between CIM objects and resources. The details are shown in Figure 1.

Based on above statements, we found two main problems about the architecture:

1. SMI-S has no specific measures to back up service information in DA. As a consequence, once received information are lost due to the failure of DA, DA has to suffer from a long information restoration time to receive information from SAs again, which is serious impact on the efficiency of storage resource discovery service provided by DA;
2. The resource management process is inefficient for UAs, in the case that multiple resources are managed. UAs have to contact these found resources sequentially to manage them. Obviously, the management processes are time-consuming in complex networks.

To overcome the two limitations, we propose two effective management schemes based on XML database and related techniques. Our major contributions can be summarized as follows:

1. We present an effective storage service information management scheme to back up service information with XML database. Through the scheme, DA can provide more efficient resource discovery service for UAs when failures occur quite often;
2. We propose another storage resource management scheme which prefetchs the resource CIM objects from least frequently managed SAs to DA and maps these objects into XML database. By this scheme, if managed SAs hit in XML database UAs can execute batch management operations on multiple storage resources to speed up management process, especially in a common case that most operation requests are read only (e.g., status queries and etc).

The remainder of this paper is organized as follows. In Section 2, the storage service information management scheme is presented. Section 3 proposes the storage resource management scheme. In Section 4, experimental results are presented to show the improved effects on SMI-S management architecture with the two schemes. We review related work about information management with XML database in Section 5. Finally, we conclude in Section 6.

## 2. Storage Service Information Management Scheme

According to section 1, Service information that DA received come from the SAs' advertisements. The advertisement is represented by SLP Service Register Message (SrvReg) [4] which is illustrated in Figure 2.

Service Location header (function = SrvReg = 3)	
<URL-Entry>	
Length of service type string	<service-type>
Length of <scope-list>	<scope-list>
Length of attr-list string	<attr-list>
AttrAuths	(if present) Attribute Authentication Blocks...

Figure 2: Service Registration Message

In Figure 2, URL-Entry represents the location of service provider, service-type defines storage resource service type and attr-list is a string encoding of the attributes of a service defined by ABNF [7]. An attribute list contains the variable number of service attributes and each attribute has one or more values. Based on the message structure, storage service information are semi-structured data. Consequently, the service information can be well saved as XML document and stored into XML database. The details of the scheme are presented in the following sections.

### 2.1. The Scheme Overview

The management scheme consists of three main components: storage engine, query engine and information manager.

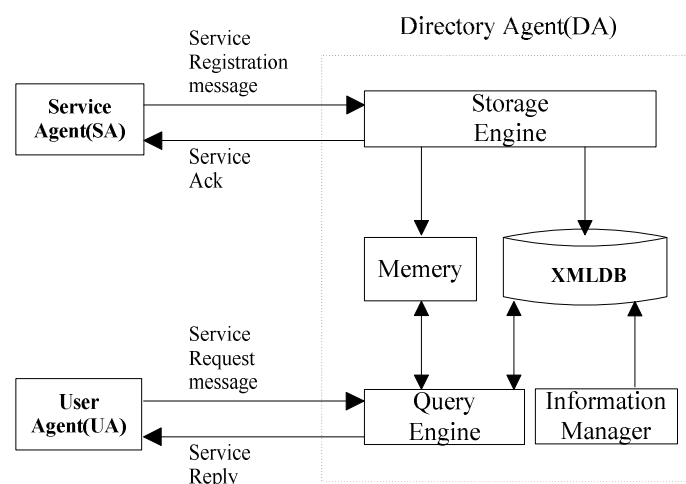


Figure 3: Storage service information management scheme

In Figure 3, Storage engine extracts service information from service registration messages and stores them in memory, meanwhile maps these information to XML documents and backs up them into XML database. Query engine fulfills user service queries. Information manager periodically accesses XML database to make statistic of service information and clean up outdated service information.

## 2.2. Storage Engine

Storage engine is used to collect, normalize and store service information into XML database. To efficiently query a lot of various types of service information in XML database, a category - based service information storage strategy is proposed in Figure 4.

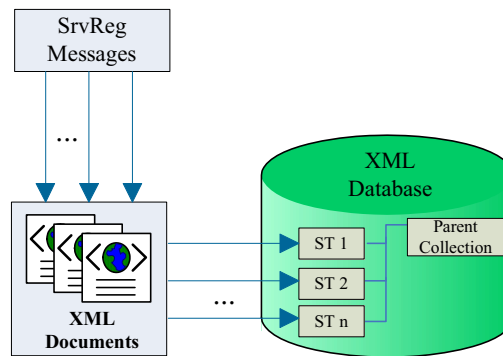


Figure 4: Category - based service information storage strategy

For XML database, XML documents are stored in a collection and are arranged into a hierarchy of collections like a traditional file system [8]. The service information mainly consists of service type and attributes list. So, we let storage engine parse the service type of the information when acquires service information from a SrvReg message. Then it attempts to establish a service type (ST) collection in XML database if the ST collection has not existed, and stores the XML document in the collection. The category storage strategy puts the same type service information together, which helps improve query performance.

## 2.3. Query Engine

When receiving service request, query engine firstly searches matched service in memory, if not found, it then queries XML database. Since almost all XML databases use XPath [9] as query language, query engine must change the requests into XPath expressions to query XML Database when receives SLP Service Request (SrvRqst) messages.

To fill the gap, we develop an efficient transformation algorithm, based on the features of SrvRqst message and XPath expression. In SrvRqst message, <service-type> and <predicate> [4] reflect the concrete service request of user. The <service-type> has a very simple form. We just treat <service-type> as one XML attribute to facilitate XPath expression. <predicate>, however, has a more complicated format that is a LDAPv3 search filter [10]. In XPath, <predicate> is also defined and has a similar filter function. But, their formats have significant differences. A few comparable examples of the two formats are shown in Table 1.

Table 1 indicates the key difference is that LDAPv3 Search Filter uses prefix notation while XPath uses infix notation between two formats. So, the kernel problem is how to realize the transformation from prefix expression to infix expression. The transformation algorithm is shown in Figure 5.

**Table. 1: Examples Comparisons of predicate formats between LDAPv3 Search Filter and XPath**

Predicate	
LDAPv3 Search Filter	XPath
(!(cn = Tim Howes))	[./cn != 'Time Howes']
(&(objectClass = Person)(sn = Jensen))	[./objectClass = 'person' and ./sn = 'Jensen']
( (objectClass = Person)(sn = Jensen))	[./objectClass = 'person' or ./sn = 'Jensen']

```

Procedure serviceXPathTransform(ST, ldapFilter)
    Input: Service Type ,ST, and ldap Filter expression , ldapFilter
    Output: Transformed Xpath expression, xpathEXP.
begin
1.   String xpath1 = ServiceType_XPath_Transform(ST);
2.   String xpath2 = LDAPFilter_Xpath_Transform(ldapFilter);
3.   Integrate xpath1 and xpath2 to form a xpathEXP;
4.   return xpathEXP;
End

Function ServiceType_XPath_Transform(ST)
begin
1.   treat ST as an attribute named @ST = '...' to form a xpathEXP;
2.   return xpathEXP;
End

Function LDAPFilter_XPath_Transform(ldapFilter)
begin
1.   int index = 0;
2.   StringBuffer strBuf = new StringBuffer(ldapFilter);
3.   while (index != the end of stringBuffer) do
4.       if ((strBuf[index] == '&') | (strBuf[index] == '|')) then
5.           op = strBuf[index];
6.           moveLeft(strBuf, index, op);
7.       else if (strBuf[index] == '!') then
8.           op = strBuf[index];
9.           changeOp(strBuf, index, op);
10.  String xpathEXP = strBuf.toString();
11.  return xpathEXP;
end
    
```

**Figure 5: Algorithm ServiceRequest\_XPath\_transform**

The transformation algorithm consists of two steps. Step 1 transforms service-type to XPath by function `ServiceType_XPath_transform`. The return is `//SA[@ST = ' ...' ]`. Step 2 changes LDAPFilter into XPath with function `LDAPFilter_XPath_transform`.

### 2.4. Information Manager

Because each service has an expiration time, information manager is responsible for cleaning up outdated service information. It will periodically travel all service type (ST) collections to check expiration time of every service information. Those expired service information will be deleted from XML database. Furthermore, it makes some statistics(e.g., least frequently managed SAs and etc) about stored service information for long-term analysis.

### 3. Storage Resource Management Scheme

Since UAs realize the storage resource management through operations on CIM objects that mentioned in Section 1, a management scheme is presented to prefetch the resource CIM objects of

least frequently managed SAs and map these objects into XML database. By this scheme, if managed SAs hit XML database UAs can batch management operations on multiple storage resources to speed up management process, especially when most operations are reading operations. Noted that to let UAs know which storage resources have been prefetched into XML database among all storage resources, a flag bit is set in service reply message shown in Figure 3. The flag with 1 means the resource has been prefetched, and vice versa.

### 3.1. The Scheme Overview

The management scheme also consists of three main components: Crawler, Adapter and Flusher.

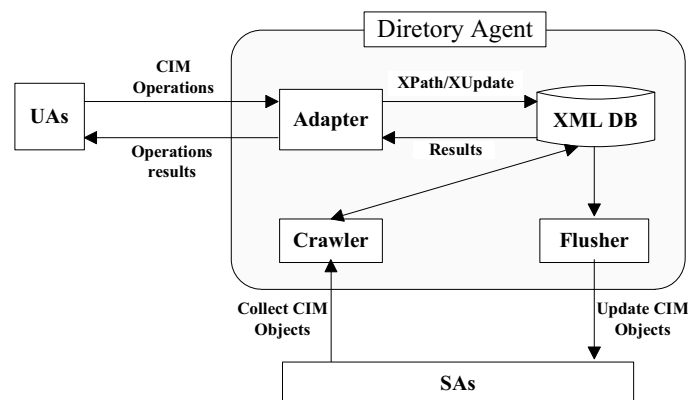


Figure 6: Storage resource management scheme

Figure 6 presents the management scheme in detail. Crawler actively collects the CIM Objects from least frequently managed SAs and keeps these CIM objects in XML database. Adapter is a bridge between CIM operations and XML operations ( XPath and XUpdate [11]). Finally, when the CIM objects stored in XML database are modified, Flusher connects related SAs to update CIM objects to keep management consistency.

### 3.2. Crawler

Crawler needs to solve two obstacles: (1) How to enumerate the least frequently managed storage resources. (2) How to map CIM Objects into XML documents.

Towards the first one, Crawler can obtain the SAs list from statistic information made by Information Manager (IM) presented in Section 2.4, because IM can learn related information about least frequently queried SAs from service replies of DA.

To solve the second obstacle, like So-Jung Lee et al. work in [12], Crawler collects CIM Objects by CIM reading operations (e.g., GetClass, EnumerateClasses, EnumerateClassnames, GetInstance and etc.) from SAs. To map the gathered CIM Objects into XML documents, we make the Document Type Definition (DTD) about the resources represented by SA in Figure 7. According to the DTD, one set of CIM Objects modelled by one SA is mapped into one XML document.

```

<!ELEMENT ServiceAgent (Location, CIM_Classes)>
<!ATTLIST ServiceAgent Dirty CDATA>
<!ELEMENT Location (IP, Port)>
<!ELEMENT IP (#PCDATA)>
<!ELEMENT Port (#PCDATA)>
<!ELEMENT CIM_Classes (CIM_Class*)>
<!ELEMENT CIM_Class(CIM_Properties, CIM_Instances)>
<!ATTLIST CIM_Class name CDATA>
<!ELEMENT CIM_Properties(CIM_Property+)>
<!ELEMENT CIM_Property (CIM_Value*)>
<!ELEMENT CIM_Value (#PCDATA)>
<!ELEMENT CIM_Instances (CIM_Instance*)>
<!ELEMENT CIM_Instance (CIM_Properties+)>
<!ATTLIST CIM_Instance ID CDATA>

```

**Figure7: DTD of SA**

### 3.3. Aapter

The Adapter has two main functions: 1. Transform CIM reading operations to XPath expression; 2. Transform CIM writing operations to XUpdate expression.

It also transforms XPath/XUpdate results to CIM operations results. Unlike ServiceRequet\_XPath\_transform algorithm presented in Section 2.3, the two transformations need no complicated format changes. For example, for CIM reading operation GetClass (name), Adapter can use a simple XPath expression `//CIM_Class [@name = '...']` to represent, with respect to CIM writing operation DeleteClass (name), it can use XUpdate expression `<xupdate: remove select="//CIM_Class[@name = '...']"/>`. Furthermore, it is necessary for the Adapter to deal with the CIM batch operation [6] which consists of multiple CIM operation requests.

### 3.4. Flusher

We borrow the idea of cache consistency strategy to keep management consistency. The attribute dirty will be set to true in SA XML document if it is modified. As a consequence, the Flusher must periodically check out the resources to flush and update them.

Meanwhile, before a dirty SA XML document is replaced, Flusher must also contact corresponding SA to flush updated information. It is noted that the modification operations on one storage resource may have correlations. That is the execution orders of these operations need be considered.

## 4. Experiment Results

In this section, we will evaluate the effects of these two management schemes on SMI-S. Our experiments show that:

1. SMI-S can provide more efficient storage resource discovery service for UAs when DA failures occur quite often, with storage service information management scheme.

2. The efficiency of storage resource management of SMI-S can be improved, through storage resource management scheme.

#### 4.1. Experimental Setup

For comparison, we establish a SMI-S prototype with JSLP [13] and WBEM Services [14]. JSLP is a JAVA-based implementation of SLP [4] and keeps storage service information with an in-memory data structure HashMap. We implement the storage resources discovery of SMI-S by JSLP. WBEM Services is also JAVA-based and provides part of SMI-S solutions that include CIMOM, CIM Client and Provider. Through WBEM Services, UAs can manage storage resources.

We choose eXist [15] to realize our schemes to enhance SMI-S prototype. eXist is a JAVA-based open source native XML database and supports XML document collection management, XPath and XUpdate.

#### 4.2. Storage Resources Discovery

We use a synthetic service information set to run this first experiment. The information set is made based on SMI-S storage service information template. In this template, the service type is “WBEM” and service attributes have 16 items including “Service-name”, “Service-description”, “CommunicationMechanism”, etc. Through template modification, we create 5,000 service information files. These files contain 20 service types from “WBEM1” to “WBEM20” and each service type has 250 service information files. The same service attributes in all service information files have different values. For example, the value of service attribute “Service-id” is from “1” to “5,000”, etc. Experimental network environment is LAN.

We estimate the efficiency of DA by the ratio of the number of success service replies to the total number of service requests in a period. In this experiment, we start 20 SAs on one host, 20 UAs on another host and run either SMI-S DA or the enhanced DA with our scheme on the third host. 20 SAs represent 20 different services selected from our service information set. Each SA sends its service information to DA at a refresh interval  $\alpha = 600$  seconds. Each UA issues a service request to DA at a refresh interval  $\beta = 20$  seconds and records the percentage of success service replies at intervals of 1 minute. All service lifetimes are longer than  $\alpha$ .

We let the two DAs reboot at the 10th min, the 30th min and the 50th min. The percentages of success service replies of the two DAs within an hour individually are shown in Figure8 and Figure9.

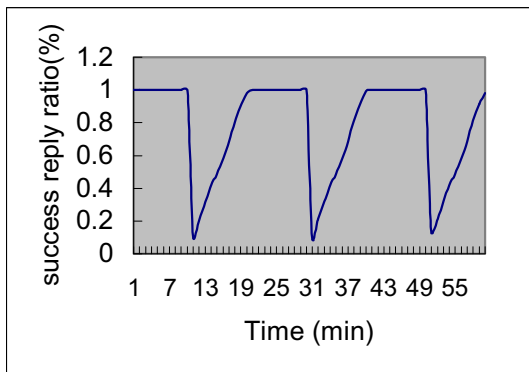


Figure 8: The success reply ratio of DA

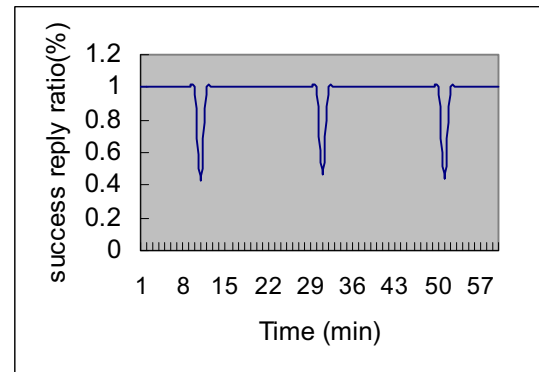


Figure 9: The success reply ratio of enhanced DA

In this test, we observe JSLP DA takes almost 10 minutes to reach 100% success reply, while enhanced DA(E-DA) only takes 1 minute to restore when failures happen. It is obviously since JSLP DA does not back up collected service information, it has to take considerable long time to restore all service information especially when the number of SAs is large. However, E-DA can take a little time to restore due to the support of eXist.

### 4.3. Storage Resources Management

In the second experiment, we deploy multiple SAs that respectively model and represent different storage resources, one JSLP DA and one UA. Experimental network environment is LAN. Given all storage resources are least frequently managed, we let the UA periodically monitor these storage resources configuration information by CIM operation EumerateInstances. We compare the management operation execution time for SMI-S and enhanced SMI-S with our management scheme. The operation execution time is shown in Figure 10.

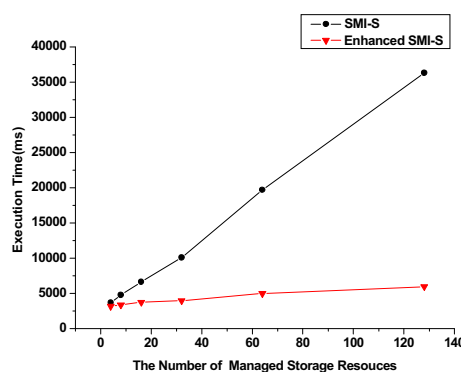


Figure 10: Manage operations execution time by the number of storage resources

We observe the execution time of enhanced SMI-S is well below the one of original SMI-S, with increasing the number of managed resources from 4, 8, 16, 32, 64 to 128. Thus the experiment shows the SMI-S management efficiency is obviously improved with our management scheme.

## 5. Related Work

In recent years, XML has become the dominant standard for exchanging and querying documents over the World Wide Web. One of the areas where XML has made its mark is in data interchange between applications or businesses, especially across the internet. XML database system is an effective approach for storing and processing XML data. Several information management applications with XML database are described in the following.

*So-Jung Lee et al.* So-Jung Lee et al. [12] propose a management system to manage the ubiquitous computing servers (U-servers) based on WBEM technologies. In [12], authors use XML database to store the management information of U-servers for long-term analysis and generate HTML documents from XML documents for Web display.

*Utz Westermann et al.* Utz Westermann et al. [16] present critical requirements for the management of MPEG-7 media descriptions. Along these requirements, authors investigate database solutions for XML documents regarding their suitability for the management of MPEG-7 media descriptions.

*Bell et al.* Bell et al. [17] describe a novel approach to business intelligence and program management for large technology enterprises like the U.S. National Aeronautics and Space Administration (NASA). One key distinction of the approach uses a “schema-less” XML database called Netmark to enable flexible integration of technology information for use by both humans and machines in a highly dynamic environment.

## 6. Conclusion and Future Work

In this paper, two effective management schemes have been proposed to improve SMI-S management architecture based on XML database and related techniques. With storage services information management scheme, we improve SMI-S to provide efficient storage resource discovery service for UAs. Meanwhile through storage resource management scheme, we enhance the management efficiency of SMI-S for storage resources. In the future, we plan to expand SMI-S to wide area network and focus on the improvement of SMI-S security mechanism.

## Acknowledgement

This paper is supported by the National Basic Research Program of China (973 Program) under Grant No.2004CB318201 and China Next Generation Internet(CNGI)(CNGI-04-5-1D).

## References

- [1] Storage Networking Industry Association (SNIA). Storage Management Initiative Specification Version 1.0.1. Retrieved October 5, 2005, from [http://www.snia.org/smi/tech\\_activities/smi\\_spec\\_pr/spec/SMIS\\_v101.pdf](http://www.snia.org/smi/tech_activities/smi_spec_pr/spec/SMIS_v101.pdf).
- [2] Distributed Management Task Force (DMTF). (Jun 14 1999). Common Information Model (CIM) Specification Version 2.2. Retrieved October 7, 2005, from <http://www.dmtf.org/standards/cim/DSP0004.pdf>.

- [3] Distributed Management Task Force (DMTF), "Common Information Model Specification Version 2.2," DMTF Specification, July 1999.
- [4] E. Guttma, C. Perkins, J. Veizades, & M. Day. (1999) .Service location protocol, version 2. RFC 2608. Internet Engineering Task Force, June 1999.
- [5] Distributed Management Task Force (DMTF), "Specification for the Representation of CIM in XML Version 2.0," DMTF Specification, July 1999.
- [6] Distributed Management Task Force (DMTF), "Specification for CIM operations over HTTP Version 1.0," DMTF Specification, Aug. 1999.
- [7] Crocker, D. & P. Overell.(1997). Augmented BNF for Syntax Specifications: ABN, RFC 2234, November 1997
- [8] Mabanza, N., Chadwick, J., & Rao, G.S.V.R.K. Performance evaluation of Open Source Native XML databases - A Case Study, The 8th International Conference Volume 3, 20-22 Feb. 2006 Page(s):1861 – 1865
- [9] Anders, B., Scott B., Don C., Mary F. F., Michael K., Jona-than R., & Jérôme S. XML. Path Language (XPath) 2.0. W3C Working Draft 30 April 2002.
- [10] Howes, T. The String Representation of LDAP Search Filters. RFC 2254, December 1997.
- [11] The XML:DB Project "XUpdate Working Draft", Technical report, 2000.
- [12] So-Jung Lee., Mi-Jung Choi., Sun-Mi Yoo., James W. Hong., Hee-Nam Cho., Chang-Won Ahn., & Sung-In Jung. "Design of a WBEM-based Management System for Ubiquitous Computing Servers", DMTF 2005 Academic Alliance Paper Competition, 2005.
- [13] SourceForge project. JSLP, Jan 2007, homepage: <http://jslp.sourceforge.net>.
- [14] Sun Microsystems, Java Web Based Enterprise Management, 2004. <http://wbemservices.sourceforge.net>.
- [15] Wolfgang M. eXist: An Open Source Native XML Database. In Erhard Rahm B. Chaudri, Mario Jeckle and Rainer Unland, editors, Web, Web-Services, and Database Systems, 2593, Erfurt, Germany, 2002.
- [16] Utz Westermann, & Wolfgan Klas, "An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions", ACM Computing Surveys, Vol. 35, No. 4, December 2003, pp.331-373
- [17] Bell, D.G. Maluf, D.A. Gawdiak, Y. Putz, P. & Swanson, K. The NASA program management tool: a new vision in business intelligence, Aerospace Conference, NASA Ames Res. Center, USA, March 2006.