Knowledge Management Framework for the Collaborative Distribution of Information

Jérôme Godard[1]*, Frédéric Andrès[1], William Grosky[2], and Kinji Ono[1]

[1] National Institute of Informatics, The Graduate School,
Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo 101-8430, Japan
{jerome,andres,ono}@nii.ac.jp
[2] University of Michigan - Dearborn,
4901 Evergreen Road, Dearborn, Michigan 48128, USA
wgrosky@umich.edu

Abstract. The management of multimedia documents, which includes storage, indexing, query optimization, and distribution, requires very strong frameworks and policies in order to ensure its efficiency and reliability. There has been much work done in the fields of databases, information retrieval, relevance evaluation, and transaction processing for any kind of data; but these four domains are too often considered separately. This definitely causes a lack of comprehensive distributed information management. We are convinced that much benefit can be obtained by apprehending a global vision of the whole management process through XML. In fact, we want to provide technologies that improve the access to heterogeneous and distributed sources of information for people sharing common interests. In order to deal with detailed and consistent information, we have to consider several layers of metadata related to users, communities, devices, and data sources. This categorized information has thereafter to be handled and efficiently used by applications combining processes from the four operative domains. Then, we claim that it is possible to offer users an appropriate viewpoint of the data, i.e. a personalized, effective, and very accurate access to the information.

1 Introduction

Knowledge management is a wide area where many domains are merging; thus, it appears to be a key issue in more and more applications. But it still lacks global approaches that consider the knowledge management from the acquisition to the dissemination, in particular for the shared information within users communities. We aim at building an Information Engine, i.e. a system offering global management services adapted to categories of users having specific behaviors and expectations. We are involved in the Digital Silk Roads project (DSR [15]), which is focused on the management of multilingual documents related to cultural aspects; it is handling all kinds of multimedia documents (including text-based, image, audio, video formats). The services to be proposed to the

* This research is partially supported by a grant (*bourse Lavoisier*) from the French Ministry of Foreign Affairs (*Ministère des Affaires Etrangères*) and the Joint Studies Program of The Graduate University for Advanced Studies (*Sokendai*).

W. Lindner et al. (Eds.): EDBT 2004 Workshops, LNCS 3268, pp. 289–298, 2004.

users by our Information Engine cover the usual database functions, user-personalized automated processes, and the management of transactions in order to ensure the capability of the system to work in heterogeneous distributed mobile environments. Personalized services are based on contexts such as localization, environmental variables (e.g. bandwidth), user's age, languages abilities, professional activities, hobbies, communities' involvement, etc. that give clues to the system about users' expectations and abilities. As text obviously is still (and indeed for many more years) the only reliable basis to build generic and portable strategies for the management of heterogeneous data, we chose to use metadata (through annotations) within XML about multimedia documents as a knowledge capture requirement. This *information about the data* has to cover four layers: users, communities, devices, and resources (which are homogeneous pieces of data, i.e. mono-type). Our goal, using well-structured knowledge management, is to precisely manipulate resources via the metadata we have about them. First, it is important to keep safely all the information we get about the resources. Then, we have to ensure the quality and the validity of the information we are storing. The third step is to properly disseminate the resources depending on the information we have about users. This approach is based on combined manual and automated processes for all the following services: annotation, storage, distributed back-up, data placement, information sharing, and relevance feedback. The main contribution of this paper is to provide a global *resource description* and manipulation framework that fits XML and enables us to enhance the collaborative distribution of information, by capturing all the knowledge that might be useful for improving the relevance of distributed semi-automated processes. In the second section, we will give an overview of the related work. Then we will present our *resource categorization model*, which allows us to identify and describe any kind of *resource*. The fourth section describes advanced services based on our model for the distribution of knowledge. In the last section, we will point out our contribution and introduce the forthcoming issues.

2 Related Work

Knowledge Distribution. The number of applications using ontologies to improve information retrieval processes and to deal with semantic heterogeneity is growing very fast; web services in particular already have several standards (e.g. DAML/OIL, ebXML) to describe service related information. Web-based Information systems have a typical structure which consists of three layers: semantic, application, and presentation. The Hera design methodology [19] considers integration and user support as aspects to be included within these three layers, which is a very relevant strategy according to us. Nevertheless, Hera uses a RDF-based ontology model which is very convenient but lacks context management support. Ontologies are also part of many semantic management frameworks for multimedia documents (e.g. audiovisual resources [18]); but most of the time, these frameworks are dedicated to a precise type of data and/or to a specific domain. Knowledge sharing is a wide area made up of many fields; moreover it covers different kinds of application, going from common memory space access to collaborative project management. It has been deeply investigated for many years and a lot of work has been produced (e.g. for software development teams [6]). As a matter of fact,

most of the ontology-based applications are influenced by initiatives for the definition of interoperable metadata standards (such as Dublin Core). We also would like to point out that XML, with its large set of tools and extensions is commonly recognized as the best framework to build ontologies. The distribution of data within communities can be partially automated in order to reduce the query workload [10]; indeed, it is possible to evaluate what kind of data might be interesting or useful for a class of user (community), and for a specific user. It is very important to choose an appropriate heuristic [12] depending on the requirements related to users and environments in order to perform data placement. Then it becomes realistic to create automated data placement processes; an example of scheduled data placement [13] indicates that much benefit can be obtain without any human interaction.

Context-Dependence. The significance of context-aware computing is dramatically increasing and promises strong improvements in human-machine interaction. Indeed, autonomous interactivity performed by an application or agents [2] can successfully be based on active and passive context-aware features [1]. But in all cases, it is important to balance the degree of autonomy in order not to bother users. According to Hess and Campbell [11], context is one of the factors that differentiates ubiquitous computing from traditional distributed computing. Many approaches for context management are available in the literature and various theories have been proposed to formalize context [5]. The range of fields using context-dependence is quite wide. It goes from very abstract analysis [4] to Artificial Life applications. Our understanding of context-dependence is slightly different as we consider contexts as dimensions; this approach has been deeply investigated for the definition of Multidimensional XML [17] (MXML) which is an extension of XML including the management of context-dependent information. Versioning, which is vital in a collaborative project, is also a context-dependent issue. In order to avoid the drawbacks of the two casual versioning schemes (store last version + backward deltas, and store all versions), an adaptive document version management scheme [3] enables the system to continuously evaluate if it is pertinent to keep each version of a document. However, this strategy does not allow us to take fully advantage of the context-dependence. A very complete versioning management of XML documents has been proposed [7] but it does not consider distribution issues. XML versioning using MXML has been defined [8] and so represents a nice opportunity to deal with versioning through XML. Another possible interpretation of distributed knowledge versioning is adaptive point of view, i.e. personalized data access. This issue is very interesting to us since it is similar to the kind of query optimization we want to provide. Giving a formal approach of a multidimensional logic, [20] defines a set of contexts with properties that seems to be very convenient for MXML.

3 Our Knowledge Management Model

Our goal is to define a generic model for the management of distributed knowledge related to any kind of multimedia document. This approach is strongly relying on XML-based annotations, which imply for users to spend a certain amount of time and to be quite precise about the information they are adding. We have the great opportunity with

DSR (which aims at creating a global repository that enables us to validate, preserve, and disseminate cultural resources) to work with more than 300 specialists in various fields who are *motivated* and *able* to annotate documents very accurately. Building such a system is a great challenge, and requires to ensure the storage, the accessibility, and the retrieval of large volumes of multilingual multimedia contents related to the silk roads through XML. In this part, we present the model we use to identify, describe, and access any kind of resources with the aim to integrate them to DSR.

3.1 Contextual Data and Model

Cultural information is very difficult to handle. Since it includes aspects such as politics, art, or history, it is impossible to consider it as a fully representative information, even when the sources are the most reliable ones. This is the kind of contextual issue we want to address. The considerable amount and diversity of information we are dealing with entails building a strong and powerful knowledge tree (ontology like) with contextual features, which fits MXML or XML namespace. We are using a *Resource Categorization Tree* (RCT) so we are able to offer a coherent model with efficient operators. The *resource* is the basic element of our model; it can be any kind of unmixed multimedia document, i.e. a monotype document (pure text, picture, video. . .). Then we add knowledge through metadata to the *resources* and obtain the atomic element of our knowledge management: *resource & annotation*. We define here our algebraic structure. Let us first give a few notations which will be used throughout the paper:

- α is a node of the tree; Ω is the set of nodes of the RCT.
- A *resource* is denoted r and is a leaf of the RCT; $\mathcal{R}$ is the set of *resources*.
- If we consider a branch of the RCT, the nodes α_{i-1} and α_{i+1} are respectively called immediate predecessor and any of the immediate successors of the node α_i. The root node of the RCT is denoted α_0.
- For any of the nodes α_i of the tree (except the leaf nodes), $\tau(\alpha_i)$ is the label of the node α_i (i.e. the label on the branch between α_i and α_{i+1}).

The primary contextual element in our approach is the *descriptor*, which brings structured and precise information about the resources:

Definition 1 (Descriptor). *A descriptor is a contextual attribute (dimension), which gives information about resources. It is denoted δ and is related to a specific node of the RCT. The set of descriptors is denoted Δ. The ordered set of descriptors of α_i is a label: $\tau(\alpha_i) = (\delta_{i,1}, \ldots, \delta_{i,p})$ where $\delta_{i,j}$ is the j^{th} descriptor of the i^{th} node and p the number of descriptors contained in label $\tau(\alpha_i)$.*

An important property of the RCT is that for any of its immediate successors α_{i+1}, the node α_i has the same label $\tau(\alpha_i)$ and so has the same set of descriptors:

$\forall i \in [0, m-1], \tau(\alpha_i) = <\delta_{i,j}>_{j=1,\ldots,p}$, where m is the number of nodes contained in the full branch (path) from the root to the leaf representing the *resource* r.

The description of the resources through the *descriptors* is integrated in the RCT in order to perform effective operations on the information stored in a community repository. This is done with the *resource categorization*:

Definition 2 (Resource Categorization). *A Resource Categorization is a branch of the RCT, which is the path extending from the root to the considered resource (i.e. leaf node). A Resource Categorization R_r of a resource r is a tuple (N_r, T_r) where N_r is the non-void ordered set of nodes of r and T_r is the ordered family of labels on N_r. The set of ordered families of labels is denoted $\mathcal{T}$.*

The deeper a label is (from the root), the more precise the information about the resource is. Since $T_r = (\tau(\alpha_0), \ldots, \tau(\alpha_{m-1}))$, where $(m-1)$ is the number of arcs the *Resource Categorization* R_r contains, we write R_r as $(N_r, \tau(\alpha_0), \ldots, \tau(\alpha_{m-1}))$ or $(N_r, <\tau(\alpha_i)>_{i=0,\ldots,m-1})$. A *descriptor* can appear in several labels of the RCT, except the *descriptors* contained in the root label $\tau(\alpha_0)$; indeed, a property of *descriptors* is that they can be used only once in a *Resource Categorization*.

All the knowledge we have about a *resource* is contained in the *resource description*; it is basically designed to structure the annotations, but it also aims at supporting the versioning of annotations:

Definition 3 (Resource Description). *A Resource Description is the complete instance of a Resource Categorization for a resource r.*

A Resource Description D_r of a resource r is a tuple (R_r, S_r) where R_r is a Resource Categorization and S_r is the ordered set descriptors values $<\sigma_{i,j,k}>$ of the resource r. It is obvious that R_r has to be equal to $(N_r, <\tau(\alpha_i)>_{i=0,\ldots,m-1})$ so we have:

$$D_r = \Big(<\alpha_i>, <\delta_{i,j}>, <\sigma_{i,j,k}> \Big)_{\substack{i=0,\ldots,m-1 \\ j=1,\ldots,p \\ k=1,\ldots,q}}$$

An extract of the RCT used for the DSR repository, and examples of labels and descriptors related to this RCT have been given in [9]. DSR *resource descriptor* list (defined by UNESCO & NII) has been influenced by the *production based attributes* of Dublin Core and by Getty's Art & Architecture Thesaurus.

3.2 Resources Comparison

Our model allows us to provide effective casual database-like operators (e.g. create, edit, insert) ; since their definition is trivial, we focus on the more complex comparative operators. The two following operators (used for the comparison of two *resources* r_1 and r_2) have in common to be made of two levels; they are first applied to sets of nodes N_{r_1} and N_{r_2}, and then, depending on this first result, to the sets of labels T_{r_1} and T_{r_2} (we use the following notation: $\tau(\alpha_i)_{r_j}$ is the label of i^{th} node of the *resource categorization* R_{r_j}):

Operator 1 (Resources Difference). *$r_1.\mathsf{diff}(r_2)$: the diff operator returns a tuple denoted (N_{diff}, T_{diff}), which is the result of a two-steps analysis. Indeed, in order to optimize the operative costs, we first check the nodes lists and notify the nodes contained in one of the Resource Categorizations R_1 and R_2 only. Then, we apply the same kind of operation to the descriptors (notation: $N^* \equiv N \setminus \alpha_0$):*

- *diff on nodes:*
  - *if* $\tau(\alpha_1)_{r_1} = \tau(\alpha_1)_{r_2}$*, then the operator returns* N_{diff} *being the list of nodes appearing only once in* $\{N_1, N_2\}$*:* $N_{diff} = N_1 \cup N_2 \setminus N_1 \cap N_2$
  - *if* $\tau(\alpha_1)_{r_1} \neq \tau(\alpha_1)_{r_2}$*, then the operator returns* N_{diff} *being the list of all nodes in* N_1 *and* N_2*:* $N_{diff} = N_1^* \cup N_2^*$*; it is possible in this case to generalize the diff operator for n resources* $(r_1.diff(r_2, \ldots, r_n))$*; thus:*

$$if \;\; \tau(\alpha_1)_{r_1} \neq \tau(\alpha_1)_{r_i} \; \forall i \in [2, n], \; then \;\; N_{diff_{1,(2,\ldots,n)}} = \bigcup_{i=1}^{n} N_i^* \subset \Omega$$

- *diff on descriptors:*
  - *if* $N_{diff} = \emptyset$*, then obviously, the operator returns:* $T_{diff} = \emptyset$
    This case implies $(N_1, T_1) = (N_2, T_2)$*, and means that* $R_{r_1} = R_{r_2}$
  - *if* $N_{diff} \neq \emptyset$*, we have to take into account a property of RCT; indeed, since a descriptor can appear in several RCT's labels, we do not only consider the non-similar labels to look for redundancies. This is the reason why we check the common descriptors between* D_1 *and* D_2*:*
    $T_{diff} = \left\{ <\tau(\alpha_i) \setminus \{\delta_{i,j}\}>, \forall \alpha_i \in N_{diff} \;\; | \; \exists \delta_{p,q} = \delta_{i,j}, \; \delta_{p,q} \in D_1 \cap D_2 \right\}$

Operator 2 (Resources Intersection). $r_1.inter(r_2)$*: As mentioned earlier, the inter operator has the same structure as diff. This time, the two-steps analysis is not required to identify different cases, but it is interesting to perform a test on the second label in order to save some processing time. The inter operator returns a tuple* (N_{inter}, T_{inter})*:*

- *inter on nodes:*
  - *if* $\tau(\alpha_1)_{r_1} \neq \tau(\alpha_1)_{r_2}$*, then obviously the operator returns:* $N_{inter} = \alpha_0$
  - *if* $\tau(\alpha_1)_{r_1} = \tau(\alpha_1)_{r_2}$*, then the operator returns:* $N_{inter} = N_1 \cap N_2$
- *inter on descriptors:* $T_{inter} = \left\{ <\tau(\alpha_i), <\delta_{j,k}>> \;\; \middle| \; \alpha_i \in N_{inter}, \right.$
  $\left. \exists \alpha_j \in N_1 \cup N_2 \setminus N_{inter} \;\; | \; \delta_{j,k} \in D_1 \cap D_2 \right\}$

From both previous operators, we evaluate the similarity between two *resources*:

Operator 3 (Resources Similitude). $r_1.sim(r_2)$*: This operator is based on the operators diff and inter (notation:* $Card(T_a)$ *is the number of descriptors contained in the ordered family of labels* T_a*); it returns:* $\rho = (\rho_N, \rho_T) \in [-1, 1]^2$
with $\rho_N = \dfrac{Card(N_{inter}) - Card(N_{diff})}{Card(N_1 \cup N_2)}$, $\rho_T = \dfrac{Card(T_{inter}) - Card(T_{diff})}{Card(T_1 \cup T_2)}$

- ρ_N *gives a global idea about the similarity between the types of* r_1 *and* r_2*.*
- ρ_T *gives a more precise evaluation about* r_1 *and* r_2 *similarity. It also allows us to find similarities between documents having different types.*

It is obvious that the sim operator provides an interesting support for advanced indexing of *resources* as it allows us to record relationships between *resources* each time a new entry is performed in the repository. We plan to add some variables in the sim operator in order to record the descriptors occurrences and then to return a weight related to the number of occurrences.

4 Adaptive Services

4.1 Environmental Knowledge

Advanced services for collaborative distribution of information must rely on knowledge related to the data (in our case through the RCT); but they also need to consider all elements that are involved in the process and might influence the access to the information. We clearly need a representation of all the useful information about the contextual entities (i.e. users, communities, and devices); this is the motivation of the profile:

Definition 4 (Profile). *A profile is a set of descriptors and values that are related to one environmental entity; it is a tuple denoted:*

$\pi = (<\delta_i>, <\sigma_{i,j}>)_{\substack{i=1,\ldots,k \\ j=1,\ldots,l}}$ *where k is the number of descriptors and l the number of ordered values for each descriptor. The set of profiles is denoted* Π.

The values can be constants (birthdate, CPU...) or variables (localization, job...); it is important to specify types in order to manage efficiently history records (a profile is time-stamped). The possible number of values for each descriptor is bounded. Moreover, the list of values for one descriptor is ordered; from the most relevant to the less one. e.g. in the case of languages, the first one must be the user's mother-tongue and then decreasingly regarding his skills.

4.2 Offering Multi-viewpoint

The word *viewpoint* has various interpretations. It can be *a perspective of interest from which an expert examines a knowledge base* [14] or *an interface allowing the indexation and the interpretation of a view composed of knowledge elements* [16]. Our approach is slightly different. Indeed, we focus on the interaction between the data and the querying environment: we use all available knowledge to extract and to provide the most relevant set of data for the user. Then, the *viewpoint* becomes the characterization of an association *resource-environment*:

Operator 4 (Viewpoint). *A Viewpoint is expressed as a function returning an ordered set of Resource Descriptions. We use* ν *to denote a Viewpoint:*

$$\begin{aligned}
\nu = \xi \circ \psi : \mathcal{R}^p \times \Delta^p \times \Sigma^p \times \Pi &\longrightarrow \mathcal{R}^q \times \Delta^q \times \Sigma^q \\
\left(<D_i>_{i=1,\ldots,p},\, \pi_e\right) &\overset{\Xi \circ \Psi}{\longmapsto} <D_k>_{k=1,\ldots,q} \\
with\ \psi : \mathcal{R}^p \times \Delta^p \times \Sigma^p \times \Pi &\longrightarrow \mathcal{R}^q \times \Delta^q \times \Sigma^q \times \Pi \\
\left(<D_i>_{i=1,\ldots,p},\, \pi_e\right) &\overset{\Psi}{\longmapsto} \left(<D_j>_{j=1,\ldots,q}, \pi_e\right) \\
and\quad \xi : \mathcal{R}^q \times \Delta^q \times \Sigma^q \times \Pi &\longrightarrow \mathcal{R}^q \times \Delta^q \times \Sigma^q \\
\left(<D_j>_{j=1,\ldots,q},\, \pi_e\right) &\overset{\Xi}{\longmapsto} <D_k>_{k=1,\ldots,q}
\end{aligned}$$

where p is the number of considered Resource Descriptions and q the number of returned Resource Descriptions ($q \leq p$), π_e *is the profile of the environment e (with* $\pi_e = \pi_u \cup \pi_d$, *u denotes a user and d a device), and* Ξ *and* Ψ *are two sets of rules:*

- Ξ *contains acceptation rules. If a descriptor value of* D_r *does not respect a rule in* Ξ*, then the set returned by* ξ *does not contain* D_r*.*
- Ψ *contains transformation rules. If a descriptor of* D_r *is involved in any rule of* Ψ*, its value might be modified depending on* π_e*.*

Each rule is a test on a pair of descriptor values; one from the *Resource Description* and the other one from the profile. Both sets of rules are deeply dependent on the type of domain the Viewpoint is applied to. ψ and ξ also allow us to rearrange in order the *Resource Description* sets by classifying decreasingly the elements respecting the larger amount of rules.

Example 1. We give here two examples for multi-viewpoint support (different from the obvious multilingual one) using the two types of rules for DSR:

- Ξ: if a *resource* of the set $<D_i>$ is a video using a codec which is not available on the user's system, then the *resource* is removed from $<D_i>$.
- Ψ: map focus points might depend on the users' location if no specific information appears in the query; in the case of DSR, querying a silk roads map would first return maps focused on the area the user is located in.

We can also consider Ψ's rules results as commands for *resources* transformation; for instance, an image, that has a bigger resolution than the one of the user's screen, would be reduced to the screen resolution. Our viewpoint definition can be seen as a query optimizer. This strategy is very useful for distributed systems and heterogeneous environments (especially mobile devices). It is of course important to define the transformation rules according to the server software environment; in the DSR case, we use some applications providing image management, text summarization... Then it becomes trivial to manage the information, and to apply the modifications depending on the descriptor values.

4.3 Data Placement

The architecture of the DSR platform deals with communities of users. The data is basically stored on a main central server, with back ups on local servers. But as most of the countries that are involved in the project have low computing and bandwidth capacities, it is important to optimize the distribution of the *resources*; this is the aim of the data placement. Indeed, using automated processes, we can dispatch efficiently and accurately the *resources* for communities and users. We have here to specify that the DSR platform has a 3-layers architecture: *servers*, *access-point*, and *devices*. Indeed, each community has a device called *access-point*, i.e. a machine that has enough computing power, storage capacity, and connect-ability to be a kind of sub-server for the other devices of the community. This architecture requires the information about a layer to be kept on the upper layer. In fact, the *server* must contain a record of all *access-points*'s profiles, and an *access-point* has details about all the *devices* and users involved in the community that the *access-point* is representing. We define the operator Dispatch (disp), which is applied to the *Resource Description* of any new *resource r* added (or updated) on the servers:

Operator 5 (Dispatch Resource). *r.disp: Let us consider the resource description of r denoted $D_r = (N_r, T_r, S_r)$, the profile of the i^{th} community $\pi_{c_i} = (T_i, S_i)$, the total number of communities denoted $\mathcal{C}$, the profile of the j^{th} user of the i^{th} community denoted $\pi_{u_{i,j}}$, and the total number of users in the i^{th} community denoted $\mathcal{U}_i$; disp is first applied on communities and then on users:*

$\forall \pi_{c_{i_{i=1,\ldots,\mathcal{C}}}}$, we define $\rho_{D_c} = \dfrac{Card(T_{inter})}{Card(T_r \cup T_i)}$, $\rho_{D_c} \in [0,1]$

- *if $\rho_{D_c} \geq s_{c_1}$, then the resource r is copied on the access-point.*
- *if $s_{c_2} \leq \rho_{D_c} < s_{c_1}$, then a link pointing on the resource r in the server is created on the access-point (index table).*
- *if $\rho_{D_c} < s_{c_2}$, then $\forall \pi_{u_{i,j}}$, we define $\rho_{D_u} = \dfrac{Card(T_{inter})}{Card(T_r \cup T_{i,j})}$, $\rho_{D_u} \in [0,1]$*
  - *if $\rho_{D_u} \geq s_{u_1}$, then the resource r is copied on the device of the j^{th} user that has the larger storage capacity.*
  - *if $s_{u_2} \leq \rho_{D_u} < s_{u_1}$, then a link pointing on the resource r in the server is created on the user local index table.*
  - *if $\rho_{D_u} < s_{u_2}$, no information is sent from the server to lower layers.*

where $s_{c_1}, s_{c_2}, s_{u_1}, s_{u_2}$ are thresholds. One of the main issues is to fix these values. We are currently defining some criteria to calculate them; by the way, we already know that the values will have to be arbitrarily adjusted after experimental results. It is also clear to us that some descriptors are more important than others; we are including a weighting factor to make the operator more effective.

Example 2. It is easy to notice the interest of disp if you consider a DSR member being an architect, and so who is part of the architect community within DSR. This person would get an easier and faster access to the data related to architecture from the access-point (e.g. many *resources* containing the buildings label). Then, according to his own profile (location, other topics of interest...), he would receive on his device some *resources* or links that are relevant to him.

5 Conclusion

This paper proposes a generic framework to address contextual problems in the case of annotated multimedia documents management within communities. We have investigated the related issues and solutions, presenting our model and showing concrete examples from DSR. Our aim is to efficiently offer the most appropriate information to the users. We introduced a proposal for a global management of distributed heterogeneous *resources* based on *categorization* and *description*, and an adaptive access model. We must point out that the services proposed are not only useful is the case of DSR; they can be applied to any collaborative project, and are promising to compensate the lack of knowledge management and drawbacks of data distribution in a P2P and mobile environment. Much work still has to be done to provide a complete framework and experimental results. We have to improve the *resources* comparison and placement operators in order to fully take advantage of our model. We also started adding transactions processing for

each layer of our architecture (using JXTA-Javaspaces-JMX). . . There is objectively no limit for services to be added; indeed, it is impossible to provide a complete management of distributed knowledge, but there are many significant improvements to be achieved with our model.

References

1. L. Barkhuus and A. K. Dey. Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined. In *Proc. of Ubicomp*, pages 149–156, Seattle, WA, USA, October 12-15 2003.
2. T. Bauer and D. B. Leake. Real Time User Context Modeling for Information Retrieval Agents. In *Proc. of CIKM*, pages 568–570, Atlanta, Georgia, USA, November 5-10 2001.
3. B. Benatallah, M. Mahdavi, P. Nguyen, Q. Z. Sheng, L. Port, and B. McIver. Adaptive Document Version Management Scheme. In *Proc. of CAiSE*, pages 46–62, June 16-18 2003.
4. M. Benerecetti, P. Bouquet, and C. Ghidini. On the Dimensions of Context Dependence: Partiality, Approximation, and Perspective. In *Proc. of CONTEXT*, pages 59–72, Dundee, UK, July 27-30 2001.
5. P. Bouquet and L. Serafini. Two Formalizations of Context: A Comparison. In *Proc. of CONTEXT*, pages 87–101, Dundee, UK, July 27-30 2001.
6. T. Chau, F. Maurer, and G. Melnik. Knowledge Sharing: Agile Methods vs. Tayloristic Methods. In *Proc. of WETICE*, pages 302–307, Linz, Austria, June 9-11 2003.
7. S.-Y. Chien, V. J. Tsotras, and C. Zaniolo. XML Document Versioning. *SIGMOD Record*, 30(3):46–53, September 2001.
8. M. Gergatsoulis and Y. Stavrakas. Representing Changes in XML Documents using Dimensions. In *Proc. of XSym*, pages 208–222, Berlin, Germany, September 8 2003.
9. J. Godard, F. Andrès, and K. Ono. Management of Cultural Information: Indexing Strategies for Context-dependent Resources. In *Proc. of DSR Symposium*, pages 369–374, Nara, Japan, December 10-12 2003.
10. S. D. Gribble, A. Y. Halevy, Z. G. Ives, M. Rodrig, and D. Suciu. What Can Database Do for Peer-to-Peer? In *Proc. of WebDB*, pages 31–36, Santa Barbara, CA, USA, May 24-25 2001.
11. C. K. Hess and R. H. Campbell. A Context-Aware Data Management System for Ubiquitous Computing Applications. In *Proc. of ICDCS*, pages 294–301, USA, May 19 - 22 2003.
12. M. Karlsson and C. Karamanolis. Choosing Replica Placement Heuristics for Wide-Area Systems. In *Proc. of ICDCS*, Tokyo, Japan, March 23-26 2004.
13. T. Kosar and M. Livny. Scheduling Data Placement Activities in Grid. Technical Report 1483, Computer Sciences Department, University of Wisconsin, USA, July 2003.
14. O. Marino, F. Rechenmann, and P. Uvietta. Multiple Perspectives and Classification Mechanism in Object-Oriented Representation. In *Proc. of ECAI*, pages 425–430, 1990.
15. K. Ono, editor. *Proc. of DSR Symposium*, Tokyo, Japan, December 11-13 2001.
16. M. Ribière and R. Dieng-Kuntz. A Viewpoint Model for Cooperative Building of an Ontology. In *Proc. of ICCS*, pages 220–234, Borovets, Bulgaria, July 15-19 2002.
17. Y. Stavrakas, M. Gergatsoulis, and P. Rondogiannis. Multidimensional XML. In *Proc. of DCW*, pages 100–109, Quebec City, Canada, June 19-21 2000.
18. C. Tsinaraki, E. Fatourou, and S. Christodoulakis. An Ontology-Driven Framework for the Management of Semantic Metadata Describing Audiovisual Information. In *Proc. of CAiSE*, pages 340–356, Klagenfurt, Austria, June 16-18 2003.
19. R. Vdovjak, P. Barna, and G.-J. Houben. Designing a Federated Multimedia Information System on the Semantic Web. In *Proc. of CAiSE*, pages 357–373, June 16-18 2003.
20. R. K. Wong, F. Lam, and M. A. Orgun. Modelling and Manipulating Multidimensional Data in Semistructured Databases. In *Proc. of DASFAA*, pages 14–21, Hong Kong, April 18-20 2001.