Information Retrieval in Distributed Environments Based on Context-Aware, Proactive Documents[1]

Michael Friedrich, Ralf-Dieter Schimkat, Wolfgang Küchlin
Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Germany

Summary

In this position paper we propose a document-centric middleware component called *Living Documents* to support context-aware information retrieval in distributed communities. A *Living Document* acts as a micro server for a document which contains computational services, a semi-structured knowledge repository to uniformly store and access context-related information, and finally the document's digital content. Our initial prototype of *Living Documents* is based on the concept of mobile agents and implemented in Java and XML.

1 Introduction: *Living Documents* and Information Retrieval

Considering multiple user perspectives, a conceptual model of information retrieval dealing with text or multi-media documents should integrate different views on documents. In addition it should allow queries addressing each of these document views separately, as well as queries for combinations. Each view is described in terms of meta data and therefore associated with the document.

Generally, meta data is data that describes other data to enhance its usefulness (Marshall 1998). We see context information as meta data describing different orthogonal aspects of the respective document. This meta data is characterized by its diversity and continuous evolvement. Incorporating such kind of meta data into the retrieval process allows to improve the overall precision because more relevant and accurate information can be comprised into the overall retrieval process.

The major problems in querying flexible attributes for meta data are the
- proprietary encoding and accessing schema for meta data of each document view.
- continuous creation and updating of meta data related to document views (temporal aspects).
- decentralization and distribution of documents' view meta data.

In decentralized communities documents should be shared, cloned and downloaded for use without a network connection. With these requirements several problems arise, like:
- Where should context information be stored?
- How can documents be processed on only temporarily connected clients?
- How can the evolution of contexts be tracked in such environments?

Our approach is to provide a general middleware component which accompanies documents during their entire document life cycle. The middleware component provides facilities to uniformly access the context information which is represented in a flexible XML-repository. Furthermore, the concept which we call *Living Documents* supports several retrieval paradigms (namely: reactive, proactive and cooperative retrieval).

1 Supported by the *Ministerium für Wissenschaft, Forschung und Kunst* of the state Baden-Württemberg, Germany. Project: Verbund Virtuelles Labor (www.vvl.de)

2 Design of *Living Documents*

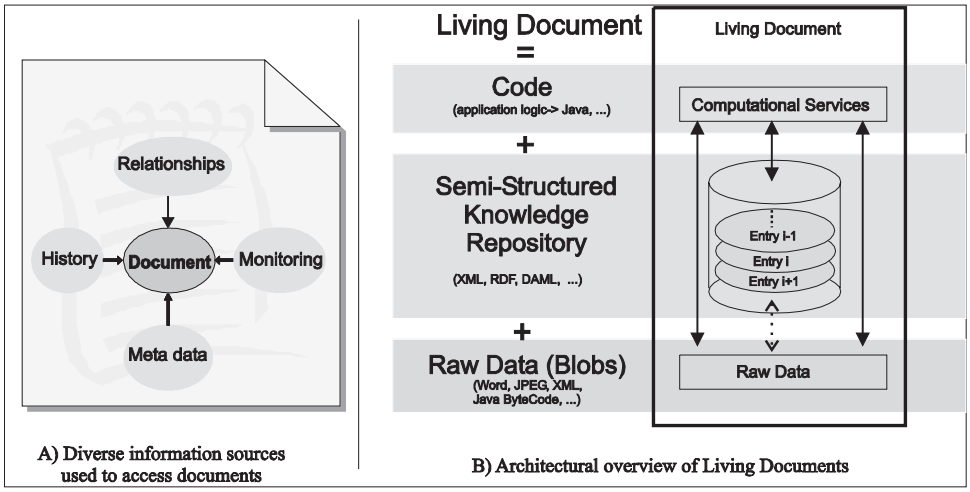

Figure 1: A) Examples for various different types of information sources which improve access to and search of documents in digital libraries. B) Living Documents are divided into three sections: Raw Data carries the documents to manage, Semi-Structured Data contains all meta data about the managed documents and the Code section keeps the computational services for accessing a Living Document and processing incoming requests (i.e. queries).

2.1 Towards a micro server architecture

A Living Document (*LD*) is a logical and physical unit consisting of three parts, as depicted in Figure 1B:

1. code or computational services
2. semi-structured knowledge repository
3. raw data

2.1.1 Computational Services

The *Computational Services* are essentially *code* fragments which provide several facilities, such as access and query capabilities or general application services. An example for such an application service is a viewing component for a document which is encoded in XML (World Wide Web Consortium (2001) . The code fragments determine the degree of activeness of a *LD* ranging from passive documents which are enriched with some arbitrary application logic to proactive documents.

The two other parts of a *LD* are accessed through the code section to ensure the integrity of the whole component. The creation and modification of the knowledge repository and the raw data and even the code itself is controlled by this layer which inhibits malicious attempts from outside to temper with the data.

2.1.2 Knowledge Repository

The *Knowledge Repository* of a *LD* provides facilities to store and retrieve the document's meta data or other related context information as depicted in Figure 1A. Each document has its own knowledge repository describing the content of the raw data section. This information is referred to as a document state information. A set of document state information builds a so-called document state report (DSR) which contains information about document behaviour, its history and reflects the contexts of this document. The context of a document reflects who uses it when, where and with which goal in mind. We present more details about contexts of *LD*s in Section .

Each DSR is encoded as a semi-structured XML document according to the *SpectoML* (Schimkat et al. 2000). Following a XML-based implementation, the generation of DSR is accomplished in an uniform way which neither does favour a particular data format nor the use of special programming or scripting languages. The use of XML as the primary data format for document state information enables a DSR with query capabilities, such as the execution of structured queries to each document state information. Therefore, a DSR builds a XML-based knowledge repository which holds all context information about the entire document life cycle.

2.1.3 Raw Data

The *Raw Data* part contains any information encoded as a digital document such as a word processing document, a music file or even serialized application code. This data is the actual information to be stored and which is described by the knowledge repository.

2.1.4 Putting it all together

The whole entity named *LD* is an atomic and self-sufficient unit which serves as a micro server for documents since it carries their logic, state and data along themselves. It can be seen as a micro edition of a huge client-server document management system for only one or a few documents. The contained meta data and logic allows the *LD* to operate in a mobile environment because every part of it is inextricably glued together. Therefore, a *LD* is equipped for only temporarily and highly distributed environments.

Finally, why is a *LD* called *living*? The state including the context of a *LD* which is represented in the knowledge repository, is characterized by its dynamic structure. Its data is growing and changing as the *LD* is used and the context of the embedded document changes. Apart from that the raw data section and even the code section of the *LD* can evolve over time. This constant change in representation and behaviour leads to the name *Living Documents*.

2.2 Implementation path

We are currently experimenting with a component architecture based on mobile agents. The aspect of mobility contributes to the notion of „living“ in *Living Documents as well*. Every *LD* is represented by an agent which can be mobile in principle. Mobile documents support the asynchronous nature of communities based on only partly connected clients like PDAs or notebooks. It allows sharing and keeping track of documents without the need of a network connection. Moreover, the agent paradigm provides a basis for real distributed communities without centralized servers and services.

We chose the mobile agent paradigm to be most flexible to distribute the *LD*s within a computer network. However, we like to emphasize that using mobile agents as basic implementation components does not restrict the concept of *LDs* to this domain. Though unlikely, other component models can be considered as more suitable in the future.

3 Implications of design

3.1 Uniform access

Living Documents provide uniform access to any kind of context-related information. Therefore, *LDs* serve as a middleware layer (Bernstein 1996, Geihs 2001) for accessing context-related information. This behaviour is essential for middleware services in general and from a rather general point of view similar to accessing relational databases uniformly by using a communication middleware component such as JDBC (Java Database Connectivity).

The uniform access of meta data is essential for *LD*s to interoperate and collaborate in groups or communities. This does neither imply that all knowledge repositories are structured the same way nor that all share the same set of access functions in the code section. The structured knowledge repository enables other *LD*s to query the stored context information in a determined way.

3.2 Context

Obviously, a document can be shared between more than one community. So the different views of a *LD* are twofold: First, each user has its view on the documents, and secondly, each community defines its own view on its subset of all documents. Consequential, *LD*s are versioned documents[2] as they change their behaviour with respect to their current context. For one person a *LD* might look as a picture document (i.e. a JPEG in the raw data section), another one might only be interested in the meta data, for instance in the author of that document. An example for such a deployment by distinct communities is different access rights. Thinking of a staff administration, everybody might see the telephone numbers but only few people are allowed to see and change the salary.

The context of a *LD* depends also on its physical location (”On which host am I?”) and on its logical location (”Which other *LD*s are nearby?”). The physical location determines the available resources like network bandwidth, memory and computing power, for example whether it resides on a PDA or a Workstation permanently connected to the Internet. The logical location affects the functionality of an *LD* if it relies on others to perform a task. As collaboration between *LD*s can improve their capabilities, one *LD* might use some specialized functions of another. This conforms to our earlier definition of a LD as a self-contained unit, because only fundamental functions for accessing LDs have to be provided. Building more complex systems out of these components goes well with the design and is strongly encouraged.

The items above affect the behaviour of a *LD* by means of general environmental conditions. The knowledge repository keeps document related information. Generally, a knowledge repository is a collection of sentences in a representation language that entails a certain picture of the world presented (Levesque & Lakemeyer 2000). For the domain of *LD*s the world is meant to be the documents’ world and the representation language is *SpectoML*. Having a knowledge repository entails being in a certain state of knowledge where a number of other properties hold. Assigning a knowledge repository to each *LD* provides several benefits for managing a DSR such as (a) easy adding of new, context-related document state information by making them dependent on the previous knowledge contained in the repository, (b) extending the existing DSR by adding new beliefs and document artifacts, (c) the possibility to explain and justify precisely the current document’s state. In our previous work (Heumesser & Schimkat 2001) we have shown how to deploy logical inference mechanisms on top of an XML markup language such as *SpectoML*.

2 Versioned documents refers to the capability of *Living Documents* to provide different and complex views on documents dynamically. It does not mean that there are different versions of one document.

3.3 Proactive *LD*s

With *LD*s acting as micro servers for documents, information retrieval can either be accomplished *reactive* in a Client-Server based way, *proactive*, as the document itself triggers a query, or *cooperative* manner, where multiple *LD*s work together to achieve a common goal. The latter paradigm can make use of a distributed environment, where the retrieval process is done in parallel by multiple components.

By deploying *LD*s the distinction between documents and applications blurs, because documents can contain (and mostly do contain) application logic which defines their behaviour. Therefore, the location of an application is hard to define during runtime. This spreading of functional components is issue of current research (Zambonelli & Parunak 2002). Application development using *LD*s follows this trend by distributing the application logic over interconnected nodes.

A proactive *LD* initiates complex tasks, discovers new services and is more than just a reactive component. For example, a *LD* can gather information over a network (e.g. web pages), process this data and continuously check the original source for updates. Another example is a persistent query for an information system, which is performed when the data source of this system changes.

3.4 Flexibility

Our document centric approach trades flexibility for performance. The basic idea is to provide a knowledge repository which is flexible and open because the set of tasks built upon this paradigm might not be predictable in advance due to the complexity and non-determinism of the documents' world and the surrounding environment. Therefore it is feasible to make the documents knowledge explicit since we might not know in advance how the knowledge will be used. Furthermore, there might be a wide variety of different context-relevant application and information sources interfacing to the documents knowledge repository. Each of them might have its proprietary application logic for dealing with the externalized *LD*s knowledge. It is viable therefore, to have a flexible, open architecture at hand when it comes to building real world applications.

4 Future Work

Initial experiments with our prototype implementation of *Living Documents* based on mobile agents show promising results. However, there are still several open issues to explore more deeply. Among these are the aspects document usage policies like sharing, cloning and distributing documents, and a thorough performance and implementation analysis.

Since the described concept of *Living Documents* is related to intensional documents as discussed by Schraefel et al. (2000), we currently explore appropriate facilities to create an active hypertext of interlinked *Living Documents* which would build a basis for further improvements of the information retrieval process within such distributed document spaces.

5 References

Bernstein P. (1996): Middleware: A Model for Distributed System Services. In: *Communications of the ACM*, Vol. 39, No. 2: p. 86 – 98.

Geihs, K. (2001): Middleware challenges ahead. In: *IEEE Computer*, Vol. 34, No. 6, p. 24—31.

Heumesser,B. D.; Schimkat, R.-D. (2001): Deduction on XML documents: A case study. In: *Proceedings of the 14th International Conference of Applications of Prolog (INAP 2001) - Stream Content Management*, p. 20 – 29.

Levesque, H. J.; Lakemeyer, G. (2000): *The logic of knowledge bases*. Cambridge, Massachusetts: MIT Press.

Marshall, C. C. (1998): Making metadata: a study of metadata creation for a mixed physical-digital collection. In: *Proceedings of the third ACM Conference on Digital libraries*, ACM Press. p. 162 - 171.

Schimkat, R.-D.; Häusser, M; Küchlin, W; Krautter, R. (2000): Web application middleware to support XML-based monitoring in distributed systems. In: Debnath, N. (Ed.): *Proceedings of 13th International Conference on Computer and Applications in Industry and Engineering (CAINE 2000)*. International Society for Computers and Their Applications. p. 203-207.

Schraefel, M. C.; Mancilla, B.; Plaice, J. (2000): Intensional hypertext. In: Gergatsoulis,M.; Rondogiannis, P. (Eds.): *Intensional Programming II*. Singapore: World-Scientific. P. 40 – 54.

World Wide Web Consortium (2001), http://www.w3.org/TR/REC-xml. *Extensible Markup Language (XML) 1.0*.

Zambonelli, F.; Parunak, H. V. D. (2002): From Design to Intention: Signs of a Revolution. IN: Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2002). To appear.

6 Contact Information

Michael Friedrich, Ralf-Dieter Schimkat, Wolfgang Küchlin
Wilhelm-Schickard-Institute for Computer Science
University of Tübingen
Sand 13
D-72076 Tübingen, Germany

e-mail: {friedrich, schimkat, kuechlin}@informatik.uni-tuebingen.de
http://www-sr.informatik.uni-tuebingen.de