Managing Digital Library Components

Hussein Suleman

Department of Computer Science, University of Cape Town
Private Bag X3, Rondebosch, 7701, South Africa
hussein@cs.uct.ac.za

Abstract. Digital library systems based on components can provide advantages such as extensibility and flexibility, but at the cost of increased complexity. High-level tools can be used to manage this complexity but only if there are appropriate machine interfaces to the pool of components. This paper discusses the facilities that were deemed absolutely necessary in order to support a wide range of independently-developed high-level functions in supporting a DL system. The common thread underlying these management interfaces suggests extensions that could be incorporated into future open interfaces for digital library components, with minimal increases in complexity, thereby maintaining the advantages of a simple component model.

1 Introduction and Motivation

Digital library (DL) systems were historically developed as monolithic applications but some recent work has focused on how instead to develop systems as collections of cooperating components [2] [20]. The shift to components is motivated by a need for code reuse and flexibility and extensibility of systems, to address an emergence of common patterns in digital library system development [6]. These components typically have well-defined interfaces and protocols for inter-component communication and communication with external systems.

The Open Archives Initiative (OAI) had a defining influence on this movement with its Protocol for Metadata Harvesting (PMH) [9], which is arguably one of the most successful standardisation activities within the DL community. The ease with which systems can be connected in the context of Web-based machine interfaces has led to other similar efforts such as RSS/Atom for content syndication [22] and SRU/W for remote searching [10].

While these protocols connect DL systems at a high level, it may be possible to also connect together components of a DL system at a lower level using similar protocols and interface specifications. This was the hypothesis of the Open Digital Libraries (ODL) project [20], which attempted to use an extended version of the OAI-PMH as the core protocol for inter-component communication. This extension, the XOAI-PMH, was intended to address some inadequacies in OAI-PMH v1.1 that have subsequently been incorporated into version 2.0 of OAI-PMH - for example, the ability to specify times at a second granularity [19]. In addition, the extension provided the ability to submit (Put) records to

a component. XOAI-PMH was designed only to allow experimentation - not as a recommendation for widespread adoption by the OAI or DL communities.

The ODL framework was used successfully in many prototype digital library systems, as well as in support of research and development of other technology. The approach of having independent components communicating over a Web protocol was evaluated for its performance and it was demonstrated that with modern Web technology (such as Java servlets) the performance was acceptable in many use cases [17].

After demonstrating that component-based systems have significant benefits, it was necessary to specifically address their shortcomings, especially the additional management needed for a system that is more complex than the old monolithic ones. Tools were created to visually design both the front- and back-ends of the DL system, stored as formal descriptions. These were then used as input to a packaging and deployment manager that would allow end-users to install and configure a component-based system without knowing of the existence of components. Finally, scalability was addressed using mobile components. In the process of developing tools, extensions to the core component model were necessary. These extensions were kept to a bare minimum to avoid losing the advantages of a simple framework.

Experiments with ODL components have further validated the core ODL model while the extensions suggest a set of component management facilities that could be available in future DL component frameworks. Thus this paper discusses extensions to the ODL framework that serve as an enabler for a wide variety of high level services with minimal extra effort. First, however, the core ODL framework is presented to provide a foundation for the discussions that follow.

2 Core ODL Framework

An ODL component is a collection of software libraries to provide a particular DL service. Examples of services provided by components are searching, browsing, metadata repositories, rating of resources and user interfaces.

In ODL terminology, a component refers only to the software libraries - in order to make use of the libraries, an instance needs to be created with a specific configuration and one or more uniquely-addressable external interfaces. The original ODL tools contain a mechanism, traditionally a command-line tool, to create instances based on a particular component. Each component can have multiple instances associated with it to provide, for example, multiple independent search services for different parts of a DL system or different sub-collections.

Figure 1 illustrates how ODL systems can be composed from individual instances of components. A common set of installed components can be used as the basis to create 2 co-located systems, with some components shared, some specific to individual systems and some replicated (possibly because of varying functionality).

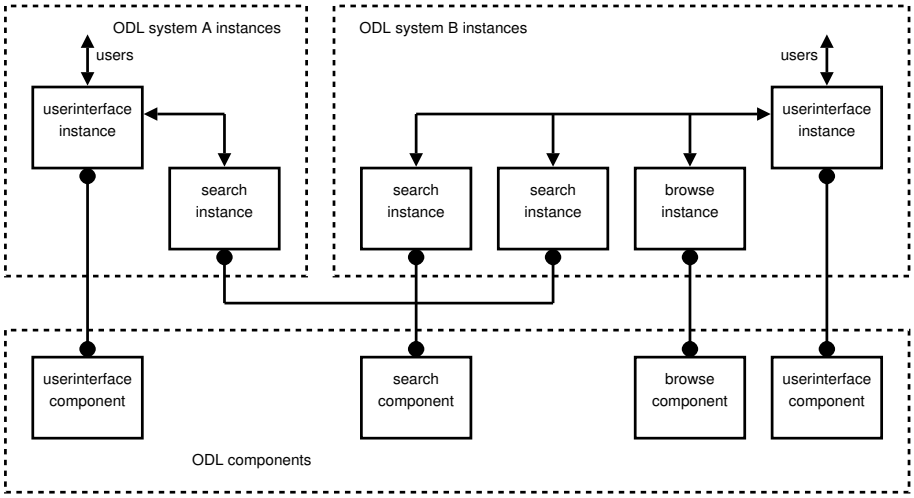

Fig. 1. Typical ODL component/instance layout

Each ODL component communicates with other components via Web-based RESTful interfaces, using protocols that were originally minimal extensions of the OAI-PMH. Thus, for example, if a user enters a search query in a Web form, the user interface instance responsible for this form submits a query to the search engine instance which executes the query and sends results back to the user interface instance, which formats the results and sends a Web page back to the user's Web browser. If the search engine instance needs to contact other instances, it may do so before responding to its query.

The ODL framework defined how components are to be developed, but there was no notion of a baseline set of services - thus individual components can be used in isolation where fully-fledged digital library systems are not needed. For example, the annotation component can be embedded into a simple website to provide a discussion forum or guestbook.

After initial experiments to validate the framework, a number of studies were conducted to provide higher level services and extend and use the components in novel ways [18]. The following sections discuss these studies and their implications for future component frameworks.

3 Component Management Extensions

3.1 Extensions of OAI-PMH

The original design of the ODL framework included a common XOAI-PMH, with higher level protocols defined for communication with individual services. This approach results in protocols that are similar and therefore easier to understand and use. In practice, if users are given a visual interface to design DL system

back-end services, where interactions can be specified and individual services configured visually, there is no need for restrictions or even recommendations on protocol use - this was verified by including popular services such as Lucene for searching and phpBB2 for discussions without the need for further protocol design [5].

As long as two components adhere to a standard for communication, whether as client-server or peer-peer, they may form part of an ODL or ODL-like system.

3.2 Management Interfaces to Components and Instances

In order to support visual management of DL systems based on components and instances, each component needs a machine interface to create new instances, modify instances and delete instances. In addition, in order to support reflection, the components must be able to list the instances associated with each.

Moore and Emslie [13] used such a set of interfaces in order to provide a DL designer with a list of existing instances to link into the systems they are designing. In addition the designer could create a new instance or modify an existing one. Mhlongo [12] leveraged the same component management interfaces in order to create digital library systems based on a formal description (saved from the visual back-end editor) after unpacking and installing the files for all necessary components. In essence, the installer would call the interface to "create instance" for each instance needed. Feng [21] used the machine interfaces purely in a read-only sense to locate instances and allow a user interface designer to link into specific service endpoints when creating custom workflows.

3.3 Global Service Discovery

At a higher level, each server needs an external Web interface that manages the list of installed components on that server.

Moore and Emslie [13] and Feng [21] both used these interfaces in order to locate components on a server based on a well-known common service endpoint.

Omar defined a further global registry across multiple machines, with communication between this global registry and the local registries at each server, operating in a manner similar to DNS [14]. Using this hierarchical service discovery system, it is possible to locate an instance based on an opaque logical name and resolve this to a physical Web service endpoint.

The individual components were changed to support redirection and resolution of logical names. The earlier mechanisms to add and delete instances were modified to now register these changes at the global registry as well.

3.4 Replication and Migration

Indirection of services makes it possible to transparently move instances from one machine to another, performing load balancing at the instance granularity.

Omar developed additional machine interfaces for instances that allow for the collection of recent execution times [14]. This makes it possible to determine

which instances are most popular and redistribute instances within a server cluster if necessary. Each component was given the ability to copy an instance to a remote location.

An independent service for balancing of load then periodically collected load information and signalled instances on overloaded servers to relocate to underloaded servers, either keeping both copies (replication) or removing the old copy after the relocation (migration).

3.5 Deployment

Deployment of DL systems is frequently ignored with preference given to end-user usability over simpler installation and administration. As most components and systems have Web interfaces, they need to be executed in the context of a Web server, acting as a Web service container. Initial ODL instances were developed as CGI applications such that they could be installed into a Web server simply by being deposited into an appropriate physical location on a server. This notion has been supported by the development of a universal Web server, which is capable of executing applications in multiple languages and server technologies - currently including Perl, PHP, Java servlets and Python - without modification or explicit installation action [11]. This universal Web server serves as an external enabler for instances that may be deployed on demand without explicit user intervention, and with no changes necessary in the component and instance interfaces.

3.6 Summary

Taking all the above applications into account, a management interface for components should include support for the following:

- For each server, service endpoints to
  - list all components on the server
  - return the total load on the server
- For each component, service endpoints to
  - list details of the protocols supported
  - list all instances
  - create, modify or delete an instance
- For each instance, service endpoints to
  - return the load of the instance
  - replicate/migrate the instance

Additional framework-level components are needed for the following functions:

- Global registry of instances
- Local registry and resolver on each server
- Load balancer

Lastly, developer tools are needed for the following functions:

- Visual design system to create formal back-/front-end descriptions
- Component packager for redeployment
- Universal Web service server

4 Related Work

The ODL components and the experimental work with management interfaces and tools related to them have been based on a strong foundation of existing and related work in other projects.

Dienst was one of the earliest service-oriented DL systems, based on early practices in Web-based machine-to-machine communication [8]. The approach taken by Dienst, whereby individual services have unique endpoints, has been carried through and has influenced ODL, OpenDLib and OAI-PMH. OpenDLib [2] is a newer component-based DL framework, which differs from ODL in a few ways, including that the initial framework contained a core infrastructure and that it contained a formal document model.

From a theoretical perspective, the DL management system envisioned by Castelli, et al., [1] provides a strong motivating basis for the extensions to the ODL framework that were created because of experimental requirements. Similarly, the 5S model [7] can lead to automatic DL generation only if DL tools have management interfaces such as those discussed in this paper.

Current DL toolkits such as EPrints, DSpace and Greenstone are considering a move towards more machine interfaces, if not necessarily components. EPrints in particular may support multiple instantiation in future [4], while Greenstone v3 [3] has well-defined Web interfaces. The Fedora system [16] is a good example of an archive component with well-defined machine interfaces for administration, but not necessarily external management.

The OAI-ORE effort [15] promises to expand on the work of the OAI and define further protocols for interoperability, which may pave the way for greater adoption of mashable DL systems, just as the OAI-PMH created the opportunity for gross system-level interoperability.

5 Conclusions and Future Work

Experiments with ODL components have demonstrated that management interfaces can enable a wide range of higher-level functionality in a component-based digital library system.

In defining a universal framework for digital library systems, it is necessary to develop a spectrum of interfaces to support not only direct use and manipulation but indirect access and machine-based manipulation.

Instead of a top-down design, these experiments have attempted to derive, bottom-up, a minimal number of extensions necessary to support many higher-level system functions. This should confirm and validate other studies that have approached the same problem from the opposite direction, that is, starting with a reasonably complete component framework such as a Grid computing toolkit.

Ultimately, the aim of this work is to derive a simple minimalist framework to support creating DL systems using a component-based approach with all the advantages of components and few or none of the disadvantages.

In future, the ODL framework needs to be redesigned to continue to be useful, in keeping with the results of these experiments and any agreed upon standards or best practices for developing simple component-based DL systems. At the same time, such future DL systems need to be developed to co-exist with other modular online systems such as learning management systems, content management systems, wikis and blogs.

6 Acknowledgements

This project was made possible by funding from University of Cape Town and NRF (Grant number: GUN2073203).

References

1. Candela, L., D. Castelli and P. Pagano (2007), A Reference Architecture for Digital Library Systems. In DELOS Conference on Digital Libraries, Grand Hotel Continental - Tirrenia, Pisa, Italy, 13-14 February.
2. Castelli, D., and P. Pagano (2003), A system for building expandable digital libraries. In Delcambre, L., and G. Henry (eds): Third ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, USA, 27-31 May, pp. 335–345. IEEE Computer Society, Washington, DC, USA.
3. Don, K. J., D. Bainbridge and I. H. Witten (2002), The design of Greenstone 3: An agent based dynamic digital library. Technical report, December 2002. Department of Computer Science, University of Waikato, Hamilton, New Zealand. Available http://www.greenstone.org/manuals/gs3design.pdf
4. EDINA (2007), the Depot. Website http://depot.edina.ac.uk/
5. Eyambe, L. and H. Suleman (2004), A Digital Library Component Assembly Environment. In Marsden, G., P. Kotze and A. Adesina-Ojo (eds): SAICSIT 2004, Stellenbosch, South Africa, 4-6 October, pp. 15–22. ACM Press, New York, NY.
6. Gladney, H. M., N. J. Belkin, Z. Ahmed, E. A. Fox, R. Ashany and M. Zemankova (1994), Digital library: Gross structure and requirements. In Proceedings of Digital Libraries '94. Available http://citeseer.ist.psu.edu/gladney94digital.html
7. Gonçalves, Marcos A., and Edward A. Fox (2002), 5SL: a language for declarative specification and generation of digital libraries. In Proceedings of Joint Conference on Digital Libraries 2002, Portland, USA, pp. 263–272.
8. Lagoze, C., and J. R. Davis (1995), Dienst - An Architecture for Distributed Document Libraries. Communications of the ACM, Volume 38, Number 4, p. 47. ACM Press.
9. Lagoze, Carl, Herbert Van de Sompel, Michael Nelson and Simeon Warner (2002), The Open Archives Initiative Protocol for Metadata Harvesting – Version 2.0, Open Archives Initiative, June 2002. Available http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm
10. Library of Congress (2004), SRU: Search / Retrieve via URL, Version 1.1, 13 February 2004. Available http://www.loc.gov/standards/sru/
11. Maunder, A., R. van Rooyen and H. Suleman (2005), Designing a Universal Web Application Server. In Bishop, J., and D. Kourie (eds): Proceedings of SAICSIT 2005, 20-22 September, White River, South Africa, pp. 86–94. ACM Press. Available http://pubs.cs.uct.ac.za/archive/00000222/01/Maunder_C13.pdf

12. Mhlongo, Siyabonga (2006), Flexible Packaging Methodologies for Rapid Deployment of Customisable Component-based Digital Libraries. MSc Thesis, Department of Computer Science, University of Cape Town. Available http://pubs.cs.uct.ac.za/archive/00000320/
13. Moore, David, Stephen Emslie and Hussein Suleman (2003), BLOX: Visual Digital Library Building. Technical Report CS03-20-00, Department of Computer Science, University of Cape Town. Available http://pubs.cs.uct.ac.za/archive/00000075/
14. Omar, M. (2007), Component-based Digital Library Scalability using Clusters. MSc Thesis, Department of Computer Science, University of Cape Town.
15. Open Archives Initiative (2007), Object Reuse and Exchange. Website http://www.openarchives.org/ore/
16. Staples, T., R. Wayland and S. Payette (2003), The Fedora Project: An Open-source Digital Object Repository System. D-Lib Magazine, Volume 9, Number 4, April 2003. Available http://www.dlib.org/dlib/april03/staples/04staples.html
17. Suleman, H. (2005), Analysis and Evaluation of Service-Oriented Architectures for Digital Libraries. In Turker, C., M. Agosti and H. Schek (eds): Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures, 6th Thematic Workshop of the EU Network of Excellence DELOS, Cagliari, Italy, 24-25 June 2004, Revised Selected Papers. Lecture Notes in Computer Science 3664, pp. 130–146. Springer. Available http://pubs.cs.uct.ac.za/archive/00000278/01/delos_2005_paper_eval_full_revised.pdf
18. Suleman, H., F. Feng, S. Mhlongo and M. Omar (2005), Flexing Digital Library Systems. In Fox, E. A., E. J. Neuhold, P. Premsmit and V. Wuwongse (eds): Proceedings of ICADL 2005, 12-15 December, Bangkok, Thailand, pp. 33–27. Springer-Verlag. ISBN 3-540-30850-4. Available http://pubs.cs.uct.ac.za/archive/00000277/01/icadl_2005_paper_revised.pdf
19. Suleman, H., and E. A. Fox (2002), Designing Protocols in Support of Digital Library Componentization. In Agosti, M., and C. Thanos (eds): Proceedings of 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL2002), LNCS 2458, Rome, Italy, 16-18 September, pp. 568–582. Springer Berlin / Heidelberg. Available http://www.husseinsspace.com/publications/ecdl_2002_paper_odl.pdf
20. Suleman, H., E. A. Fox, R. Kelapure, A. Krowne and M. Luo (2003), Building Digital Libraries from Simple Building Blocks. Online Information Review, Volume 27, Number 5, pp. 301–310. Emerald Publishing. Available http://pubs.cs.uct.ac.za/archive/00000013/01/oir_2003_oaiodl_revised2.pdf
21. Suleman, Hussein, Gary Marsden and Fu-Yao Feng (2006), Customising Interfaces to Service-Oriented Digital Library Systems. In Sugimoto, S., J. Hunter, A. Rauber and A. Morishima (eds): Proceedings of 9th International Conference on Asian Digital Libraries (ICADL 2006), 27-30 November, Kyoto, Japan, pp. 503–506. Springer-Verlag. Available http://pubs.cs.uct.ac.za/archive/00000327/01/icadl_2006_carl.pdf
22. Winer, Dave (2002), RSS 2.0 Specification. Berkman Centre for Internet and Society. Available http://blogs.law.harvard.edu/tech/rss