Moving Digital Library Service Systems to the Grid

Leonardo Candela, Donatella Castelli, Pasquale Pagano, and Manuele Simi

Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" - CNR
Via G. Moruzzi, 1 - 56124 PISA - Italy
{candela, castelli, pagano, simi}@isti.cnr.it

Abstract. The architecture of a digital library service system strongly influences its capabilities. In this paper we report our experience with the OpenDLib system, which is based on a service-oriented architecture, and we describe how, in the attempt to better satisfy the user requirements, we decided to develop a digital library service-oriented infrastructure on Grid. We also briefly introduce this infrastructure and present the open issues related to its development.

1 Introduction

Four years ago, the DLib group at ISTI-CNR began to develop a Digital Library Service System (DLSS), i.e. a system for creating and managing digital libraries (DLs). In designing this system, named OpenDLib [2], our aim was to create a customizable system that, when appropriately configured, could satisfy the needs of different application frameworks.

Our first step in designing this system was to clarify what we meant for DLs and to identify the features that a system able to implement DLs should provide. This analysis brought us to understand that a DLSS is a complex system that must not only offer powerful user functionalities (e.g. search, browse, annotation) but it must also implement basic functions for supporting the fruition of the user functionality and for guaranteing the quality of the overall DL service, e.g. its availability, scalability, performance. Moreover, it must satisfy a number of other desiderata, like be extensible, easy to install and to maintain.

In order to create the conditions for achieving the required level of quality we analyzed a range of possible system architectures and, finally, we decided to adopt a distributed, dynamically configurable, service-oriented architecture (SOA). The development of a DL system based on this architecture required a greater implementation effort with respect to the development of centralized system since a number of services dedicated to the co-ordination, management and optimal allocation of the different service instances had also to be provided. OpenDLib is now an operational system that has been used for building a number of DLs [13,14]. Each of these DLs has its own specific distributed architectural configuration that reflects the needs of the application framework where it operates. In all these different frameworks the distributed service architecture has

proved to be a valid instrument to satisfy a number of requirements that could not have been met otherwise. The greater development effort has thus be highly compensated by the better quality of the DL functionality exposed to the users.

New architectural approaches have emerged, or have been consolidated since we designed the OpenDLib system, e.g. Web services [4,12], P2P [11], Grids [9,10]. These approaches provide features that simplify the implementation of a distributed DL architecture and offer a number of new possibilities for implementing novel user functionalities and for enhancing the quality of the overall DL.

The DLib group at ISTI-CNR, with a number of other European research organizations and software companies, have recently set up a project, DILIGENT, for creating a DL infrastructure that exploits these new approaches. In particular, our plan is to built DILIGENT as a service-oriented application of the Grid infrastructure released by the EGEE Project [6].

In this paper we introduce the OpenDLib architecture by focusing on the aspects related to the services management and we report our experience in operating this system. We also describe why we decided to move towards the new DILIGENT architectural framework, what we expect from it, and the open research issues that must be addressed before starting its development.

The rest of this paper is structured as follows: Section 2 introduces the main requirements that motivated our choice of a distributed service architecture for OpenDLib; Section 3 describes the OpenDLib architecture and the lessons learned while developing and using it; Section 4 briefly introduces the DILIGENT DL infrastructure by describing its architecture and highlighting its open issues; finally, Section 5 concludes.

2 Requirements for a digital library system

The objective of the OpenDLib project was to create a software toolkit that could be used to set up a digital library according to the requirements of a given user community by instantiating the software appropriately and then explicitly submitting new documents or harvesting the content from existing sources.

The design of the system was strongly influenced by the requirements collected from some potential user communities. In particular, the requirements that mostly influenced the OpenDLib architectural choices are:

1. There is a set of core DL functionalities, such as search, retrieval, access to information objects, that any DL should provide. The format in which each of these functionalities is presented to the user is usually different since it complies with the application specific vocabularies and rules. In addition to the core functionalities, each DL, usually, *must provide other specific functionalities for serving the application-specific requirements.*
2. New organizations may ask to participate to the DL during its lifetime and additional functionalities may be required to satisfy new needs. *A DL must be able to dynamically evolve by adapting itself to these new situations.*
3. The handling of a DL can be expensive in terms of financial, infrastructural and human resources. Many organizations are confident that this problem

can be overcome by adopting a *DL federated model*. According to this model, multiple organizations can set up a DL by sharing their resources without losing, if required, control over their own resources. For example, they can store their information objects locally or host key services on their computers.

4. *The users of a DL require a good quality of the service (QoS)*, i.e. an acceptable level of non-functional properties such as performance, reliability, availability and security.
5. *Access to content and services is usually regulated by policies*. These can specify, for example, that a collection of objects is only visible to a particular group of users, or that a set of services can only be used free of charge for a given time interval.

In order to satisfy the above requirements, we decided to adopt a service-based architecture, i.e. an architecture in which all the functionalities are defined as independent services, with well-defined invokable interfaces, that can be called in defined sequences to form business processes. Following this architectural paradigm, we implemented OpenDLib as a federation of services, each of which implements a well defined functionality.

The next section describes the DL architectural framework that we designed following this choice.

3 The OpenDLib architectural framework

The OpenDLib architectural framework consists of an open and networked federation of cooperating services[1]. These services cooperate, via message exchange, in order to implement the OpenDLib functionality. In particular, the federation comprises a number of services that implement the user functionality. These services, collectively called *application service*, are listed in Table 1.

The communication among services is more complex than a simple client-server application. A service can act both as a provider and as a consumer, and use relationships may exist a priori among any subset of the services. In fact, services can be combined in different ways to support different functionality, and the same services may be used in different ways, in accordance with the restrictions placed on its use and the goal of use.

The communication among services is regulated by the OpenDLib Protocol (OLP) [3]. Such protocol imposes a set of rules that a service must follow in order to communicate with the other services belonging to the federation. OLP requests are expressed as URLs embedded in HTTP requests. All structured requests and responses are XML-based.

The OpenDLib services can be centralized, distributed or replicated on different hosting servers. An OpenDLib DL thus usually comprises multiple instances

[1] Hereafter, with the term "service" we mean an OpenDLib software module that supplies a certain task via a well-defined interface. Each module is able to communicate with other OpenDLib modules and can be deployed and hosted on a server.

Service name	Main performed tasks
Repository	Storage and dissemination of documents that conform to a document model termed DoMDL [1]. These documents can be structured, multilingual and multimedia.
Multimedia Storage	Storage, streaming and downloading of video manifestation of a document, dissemination of videos either as whole documents or as aggregations of scenes, shots and frames.
Library Management	Submission, withdrawal and replacement of documents. It is configurable with respect to the metadata formats accepted.
Index	Document retrieval parametric w.r.t. the metadata format, the set of indexed fields, the result set format and the query terms language.
Query Mediator	Document retrieval by dispatching queries to the appropriate Index service instances and by merging the result sets, taking into account the peculiarities of available Index instances.
Browser	Construction and use of appropriate data structures, termed indexes, for browsing the library content, parametric w.r.t. the metadata formats, the set of browsable fields, and the result set format.
User Interface	Mediations among human user and application services.

Table 1. OpenDLib application services.

of the same service type hosted on remote servers of different organizations. This distribution provides an appropriate context for supporting the DL federated organizational model and for ensuring quality attributes such as performance and scalability.

All the OpenDLib services of the current release are highly configurable. It is possible, for example, to select the accepted metadata formats of an Index, the query language of a Query Mediator, the publishing and service hosting institutions, the number of service replica, etc. This provides a great flexibility and permits to use the system in a variety of different DL application frameworks.

OpenDLib supports three kinds of dynamic expansions: 1) new services can be added to the federation; 2) new instances of a replicated or distributed service can be mounted on either an existing or a new hosting server; 3) the configurations of the services can be modified so that they can handle new document types, new metadata formats and support new usages. By exploiting these kind of expansions new application specific needs can be easily satisfied.

The management of a dynamic, customizable, independent set of services aimed at ensuring the desired quality of service is not trivial. It involves many functions such as: *security*, e.g. authorization of the request, encryption and decryption as required, validation, etc.; *deployment*, allowing the service to be redeployed (moved) around the network for performance, redundancy for availability, or other reasons; *logging* for auditing, metering, etc.; *dynamic rerouting* for fail over or load balancing and *maintenance*, i.e. management of new versions of the service or new services to satisfy new users needs.

In the rest of this section, we focus our attention on these services management aspects presenting the approach that has been taken in OpenDLib. In particular, we describe the two elements that together supply a "basic infrastructural layer" for supporting an optimal co-operation among the distributed services: the *Manager Service*, which maintains a continuously updated status of the networked federation of services, checks its consistency and controls the flow of the communication, and the *OpenDLib kernel* module that implements the rules imposed by the protocol embedded in each application service.

3.1 The Manager Service

We have seen that OpenDLib supports different kinds of "on-the-fly" expansions. Most of these expansions, even when they regard a specific service instance, also require a change in the configuration of other instances in order to allow them to consider the new characteristic of the service. For example, when a Repository instance is modified to accept a new metadata format, at least one Index instance must be updated to index the new format; when a new Query Mediator instance is set up to reduce the workload on the existing Query Mediators, then a certain number of User Interfaces must change their communication flow and address their service requests to the new instance.

Similar updates are needed when the conditions of the underlying network change, e.g. when there is a network failure, when the number of requests sent to a service instance exceeds an established threshold.

All the above updates in the configuration of the federation are controlled by the Manager Service, which following established algorithms, always derives the best routing strategy required to achieve a good QoS.

The Manager service is partially configured by the DL administrator at the start-up of the system. Its configuration parameters contain the minimum information required to specify the DL topology: e.g. the address of the hosting servers; the list of the services and whether they are centralized, replicated or distributed; the number of instances for each service; their allocation to the servers, etc. Configuration parameters contain also a number of consistency rules that specify the legal configurations of the service instances in the federation. These rules strictly depend on the type of service and on the "use" relation that links them. For example, the language of the terms in a query that is processed by the Query Mediator must be one of the languages indexed by the used Index services, the document and metadata descriptions submitted to a Library Management Service must conform to those managed by the corresponding Repository.

By exploiting the information about the DL architecture acquired at start-up time, the Manager begins to collect more detailed information about the service instances by periodically sending them appropriate protocol requests. It then processes the information gathered, controls its consistency, and takes decisions about the organization of the federation, such as, for example, the communication paths among the services instances. These decisions can change over time according to the value of different parameters, such as the set of running service instances, the workload of a server and the status of the connection.

All service instances notify the Manager of any changes in their configuration and on the status of the service federation. The Manager updates the architectural map and executes the necessary steps to collect information about any new instance. The service instances periodically harvest information about the federation from the Manager. For example, each service that uses another replicated service asks for the address of the instances that can serve its requests.

The instances of the federation can configure themselves either by directly exploiting the information received from the Manager, or by sending appropriate service requests to the instances whose addresses have been obtained through the Manager. Once configured, the various instances can start the co-operation required to process the DL user requests.

3.2 The OpenDLib kernel

When designing and implementing a service-oriented architecture, a common task is to provide an high level abstraction of the communication among the distributed services. This point have been carefully planned in the OpenDlib system.

The OpenDLib kernel is the basic software layer where the OpenDLib services are built on and it includes tools and software modules that allow a service to use the OpenDLib Protocol and to speak to each other no matter what is their physical location.

In order to meet these needs, a small messages dispatcher module is installed on each OpenDLib network node. This module is in charge to dispatch the incoming messages to the appropriate service hosted on the node. When a service sends a request to a remote service, it actually interacts with the dispatcher module of the target node. Afterwards, the request is interpreted and converted in a manner that a generic OpenDLib service is able to manage.

Another main step in this direction is represented by the *ProtocolManager* module. This part of the kernel is responsible to build a service request according with the rules of the protocol. Moreover, it allows to interact both with remote and local services in a transparent way. In order to achieve this second functionality a careful design of the services interface has been conducted. The study produced a solution that facilitate the conversion of a network message in a local one if the producer and the consumer of the message are hosted on the same node.

Finally, the kernel provides a set of utilities that simplify the management of the configuration and status of the federation of services disseminated by the Manager Service.

Major results of these efforts are:

- if the communication needs to be changed for any reason, the change is localized in some specific points only;
- an abstraction layer is provided, so new services can be added more easily, avoiding to deal with communication details;
- a mechanism that permits an optimization of the network traffic among the different nodes is provided.

3.3 Lesson Learned

OpenDLib is now a running system. A number of DLs [13,14] have been built using it and several others are under construction. The experience made during this experimentation has validated some of our design choices. In particular, we now firmly believe in the power of a service-oriented architecture for DLs. This type of architecture provides an extensible framework where application specific services can be added to satisfy the local requirements. Moreover, by supporting a federated maintenance model, it naturally supports the creation of large DLs obtained by dynamically and progressively aggregating resources provided by many disperse small organizations.

The experience made so far, however, has also highlighted a number of weakness of the DL system we have built. In particular:

- In the four years that have been spent in implementing and experimenting OpenDLib both the enabling technologies and the standards evolved. Today, web services and SOA are highly diffused and quite *standard* solutions for web applications. Many specifications dealing with common issues of this framework, e.g. security, resource description, etc., springs up in the WS-* family. Grid technology, offering also computational and storage power on demand, is now meeting these technologies with the WSRF. P2P technologies, popularized by (music) file sharing and highly parallel computing applications (e.g. SETI@home), can be employed successfully to reach some design goals such as scalability, availability, anonymity. Certainly, these new standards and technologies provide more powerful and advanced solutions than the ad-hoc one originally implemented by OpenDLib.
- Multimedia and multi-type content information objects are emerging as alternative and more powerful communication vehicles. In OpenDLib, the handling of these objects is limited since, e.g. multimedia documents may require high computational capacity to be opportunely and fully managed and employed.
- Many of the user communities that demand for DLs are small, distributed, and dynamic; they use the DL to support temporary activities such as courses, exhibitions, projects. Even if a digital library service system like OpenDLib can be used to setting a DL reducing the cost, these costs are still too high for certain communities of users. Each time a new DL is created appropriate computer and storage resources are needed for hosting the OpenDLib services. Furthermore, specialized technical staff with appropriate skills is required to configure, install and maintain the system.

Certainly, the recent advances of the technology makes now possible to propose better solutions to the requirements presented in Section 2. With this purpose in mind and in order to overcome the above limitations, our group, with other European research and industrial organizations, has recently set up a new EU FP6 integrated project, DILIGENT, that will start its activity on September 2004. The aim of this project is to design and experiment the development of a new SOA DL infrastructure on a Grid enabled technology. The objectives of this project and the expected results are briefly introduced in the next section.

4 The next step: DILIGENT

The main objective of the DILIGENT project is to create a knowledge infrastructure that will allow members of dynamic user communities to build-up *on-demand transient virtual DLs* (VDLs) that satisfy their needs. These DLs will be created by exploiting shared resources, where by resources we mean content repositories, applications, storage and computing elements.

The DILIGENT infrastructure will be an evolution of a DLSS. By exploiting a wider notion of sharing, this infrastructure has the potential of reducing the cost of setting up DLs thus enabling a larger adoption and use of these systems.

The DILIGENT infrastructure will be constructed by implementing a number of DL services in a Grid framework. In particular, DILIGENT will exploit the efforts of the EGEE project [6] by relying on the Grid infrastructure that will be released at the end of this project. We expect that by merging a service-oriented approach with a Grid technology we will be able to exploit the advantages of both. In particular, the Grid should provide a framework where a better control of the shared resources is possible. Moreover, it should enable the execution of very computational demanding applications, such as those required to process multimedia content.

Our current plan is to develop all the DILIGENT services following the rules established by the new Web Service Resource Framework (WSRF) [5,7]. This framework, which comprises six Web services specifications defining the *WS-Resource approach* to modelling and managing state in a Web services context, is very suitable to satisfy the needs of a DL application where both stateless and statefull resources cohabit.

The services that will be developed by the DILIGENT infrastructure project can be logically organized into three main layers:

- *Digital Library Layer*. This layer consists of a set of reliable and dependable production-quality services covering the core functionalities required by DL applications. This set provides submission, indexing and discovery of mixed-media objects (documents, videos, images, environmental data, etc.), and the management and processing of these objects through annotation, composition, cooperative editing, etc. It also supports the dynamic creation and access to transient VDLs. Each service of this area will likely represent an enhancement of the functionalities provided by the equivalent non-Grid-aware service as it will be designed to take full advantage of the scalable, secure, and reliable Grid infrastructure.
- *Application-Specific Layer*. This layer contains the set of services provided by users communities that have decided to share their legacy content and application-specific resources.
- *DILIGENT Collective Layer*. This layer is composed by services that enhance existing Grid collective services, i.e. those global services needed to manage interactions among resources, with functionalities able to support the complex services interactions required by the Digital Library Layer.

The services of the Collective Layer (see Table 2) and those provided by the Grid middleware, play the same role of the "basic infrastructural layer" in OpenDLib as they manage the federation of services and resources that implement a VDL. In this environment, however, the management is more complex since the set of servers where the DL services are hosted is not known a priori and it can vary dynamically during the VDL lifetime.

Service name	Main performed tasks
Information Service	Discovering and monitoring of a set of distributed resources, enabling other services to be aware of the environment, or part of it, that hosts them. By maintaining a *real-time* monitoring of the whole set of available resources, single services and VDLs can be enabled for self-tuning resource usage and workload-balancing maximizing the use of available resources.
Broker & Matchmaker	Optimal distribution of services and resources across Grid nodes, by promoting efficient usage of the infrastructure through the identification of the *best* Grid node where to allocate a service or a resource.
Keeper	Bringing together the set of services belonging to a VDL by assuring the QoS characteristics required. Create, address, inspect and manage the lifetime of all the resources needed to satisfy the definition criteria of the library.
Dynamic VO Support	Creation of the Grid *operational context* associating users, user requests and set of resources that enables VDLs to work accordingly with the resource sharing policies and agreements.

Table 2. DILIGENT Collective Layer services.

This new DL architecture infrastructure, at least from the theoretical point of view, provides the support required to meet the requirements that we have identified previously and to overcome the limitations that we have experimented with OpenDLib. However, combining concepts and techniques belonging to different research fields and disciplines (DL, Information Retrieval, Grid, data management, Web Services, information systems, P2P, etc.), the DILIGENT project has to solve at least the following open issues:

- Sharing of resources is acceptable only if it is highly controlled. Strong and valid security and policy mechanisms that take into account the rules of the DL providers and consumers must be developed.
- In order to dynamically create a VDL the system must be able to automatically select and retrieve the resources that better match the demand of the library creator. This requires an appropriate description of the DL resources and powerful discovery mechanisms.
- The SOA approach proposed for the DILIGENT infrastructure defines only the higher level structure of the architecture. Each service belonging to the library can internally adopt a different architecture, e.g. an Index can be

realized both using a P2P technique as well as a centralized service. This heterogeneity certainly complicates the digital library management and demands for more sophisticated algorithms.
– The different architectural framework in many cases may suggests better algorithms for implementing the DL functionality. For example, a Grid framework, which offers a high computing capacity, may enable the exploitation of complex but powerful algorithms which were not realistically possible in the traditional DL frameworks. One of the challenges in DILIGENT will certainly be to identify new algorithms that exploit the capabilities of this new DL framework.

5 Conclusion

The paper describe the architecture of the OpenDLib DL service system and it introduce the main architectural plans for the DILIGENT knowledge infrastructure.

In carrying out this experience we have learnt that the realization of a DL system not only requires the implementation of the functionality that are directly perceived by the users but it also requires the development of what we called a "basic infrastructural layer". In this paper we mainly discuss this layer from the point of view of the management of the DL services, but there are other important functionalities that a basic infrastructure layer should provide, like independence from the physical organization of the content, policy control, etc. The extent of the basic infrastructure layer strongly depends on the underlying architecture choices. The more distributed, flexible, dynamic, customizable the architecture framework is, the greater is the layer required to ensure the quality of the DL services.

In OpenDLib this layer has been constructed from scratch, by adopting ad-hoc solutions. At the present, however, more standards and powerful solutions are possible. With the DILIGENT project, our challenge is to build a new DL service system that takes into account these new openings and to contribute to the evolution of the architectural infrastructure for future DLs.

References

1. Donatella Castelli and Pasquale Pagano. A flexible Repository Service: the OpenDLib solution. In J. Á. Carvalho, Arved Hübler, and Anna A. Baptista, editors, *Proc. of the* 6^{th} *International ICCC/IFIP Conference on Electronic Publishing*, pages 194–202, 2002.
2. Donatella Castelli and Pasquale Pagano. OpenDLib: A Digital Library Service System. In *Proceedings of the* 6^{th} *European Conference on Digital Libraries (ECDL2002)*, pages 292–308. Springer-Verlag, 2002.
3. Donatella Castelli and Pasquale Pagano. The OpenDLib Protocol. Technical report, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, 2004.
4. Ethan Cerami. *Web Services Essentials (O'Reilly XML)*. O'Reilly & Associates, 2002.

5. Karl Czajkowski, Donald F. Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke, and William Vambenepe. The WS-Resource Framework. *White paper*, 2004.
6. EGEE Team. EGEE: Enabling Grids for E-science in Europe. http://public.eu-egee.org.
7. Ian Foster, Jeffrey Frey, Steve Graham, Steve Tuecke, Karl Czajkowski, Donald F. Ferguson, Frank Leymann, Martin Nally, Tony Storey, William Vambenepe, and Sanjiva Weerawarana. Modeling Stateful Resources with Web Services. *White paper*, 2004.
8. Ian Foster and Adriana Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Revised Papers*, volume 2735, pages 118–128, 2003.
9. Ian Foster, Carl Kesselman, Jeffrey Nick, and Steve Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.
10. Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.
11. Nelson Minar, Marc Hedlund, Clay Shirky, Tim O'Reilly, Dan Bricklin, David Anderson, Jeremie Miller, Adam Langley, Gene Kan, Alan Brown, Marc Waldman, Lorrie Cranor, Aviel Rubin, Roger Dingledine, Michael Freedman, David Molnar, Rael Dornfest, Dan Brickley, Theodore Hong, Richard Lethin, Jon Udell, Nimisha Asthagiri, Walter Tuvell, and Brandon Wiley. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.
12. Eric Newcomer. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, 2002.
13. OpenDLib Team. e-Library: a public OpenDLib instance. http://elibrary.isti.cnr.it.
14. OpenDLib Team. The ARTE Library: an OpenDLib instance. http://odl-server1.isti.cnr.it.