# Literature review on Archiving strategies and architecture

Craig Stevenson†
Department of Computer Science
University of Cape Town
Cape Town South Africa
stvcra005@myuct.ac.za

## ABSTRACT

The internet is home to a vast amount of knowledge, much of which we do not have control over. This means that at any given time a knowledge source, one depends on, may suddenly vanish. Reasons for this wars, natural disasters or simply funding for a specific knowledge source running out. In order to combat this We intend to build a digital library that will store other digital libraries(knowledge sources) in such a way that (1) digital resources will be preserved in the long term, (2) changes to digital resources will be recorded and stored to produce versions of those digital resources and (3) we shall provide offline functionality for the digital library such that content will be preserved through crashes, as well as providing areas with poor network connections.

## Introduction

This document presents a literature review examining various techniques to build a digital library. The main purpose of this review is to gain some understanding of the technologies out there and to evaluate them, so as to assist in the design of the final digital library.

This review will include examining various digital library architectures and repository architectures to get an understanding of the high level design of the system, it will then look at already available toolkits that can help simplify the process of creating a digital library after an architecture is chosen. The review will also examine various client side(offline) techniques and tools for building digital library services and then finally it will look at web-archiving in particular examining the various crawler technologies out there, since this will be one of the primary techniques for acquiring digital objects for our archive.

.

## CCS CONCEPTS

• Digital libraries, searching, web-archiving

## KEYWORDS

Archiving, Architecture, Web-archiving , DL toolkits

## 1 Digital Library architecture

A digital library is a complex system, hence if one is to build a rigorous and functional digital library, appropriate architecture needs to be followed. In order to choose an appropriate architecture the following properties of architecture need to be considered: whether the architecture is service driven, is open, scalable, preserves data long term, Privacy, Practical and supports Modularity [1].

According to[1], components of a digital library, at a high level, can be categorised into these four categories: user Interface, repository, handle system and search system.

### 1.1 Layered Architecture

One of the most common architectures that satisfies this is a layered architecture. This architecture separates the functions of Insertion, storage, management and consumption into layers[2]. Management being a layer that oversees the repository since that is the most complex part of a digital library.

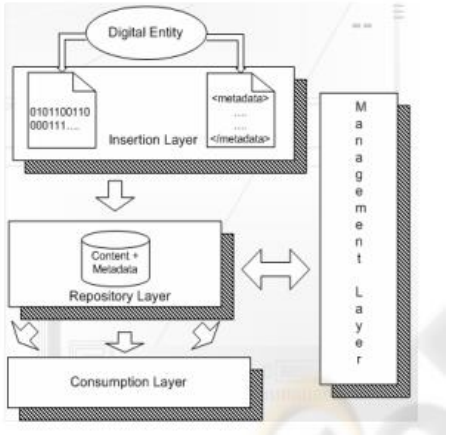


**Figure 1: Layered Architecture diagram**

### 1.2 Agent based architecture

Another architecture is an "Agent based architecture" [1], which is a more decentralized approach to building a digital library. This architecture uses autonomous entities called agents to perform its functions. Agents specialize in specific services that a digital library performs and will only perform those services. Agents work together by "negotiating" with other agents in order to gain access to the resources they require[3].

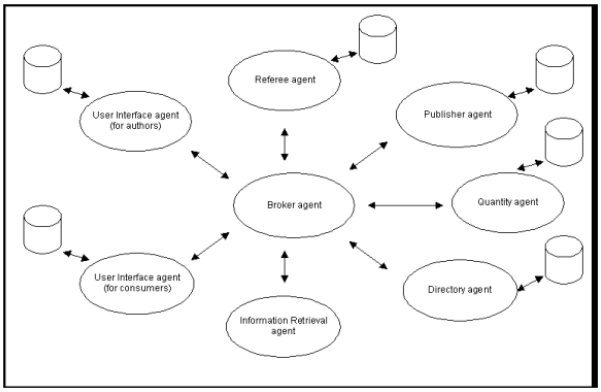


**Figure 2: Agent Based Architecture Diagram**

### 1.3 Open digital library architecture

The final architecture that will be discussed is the Open Digital Library Architecture[4]. This is a component based architecture which relies on other digital libraries exposing their meta data which is then used by the digital library requesting that information. This meta data is communicated by using the OAI-PMH. ODL's take this a step further, by having digital libraries it communicates with also be providers of certain services digital libraries provide.

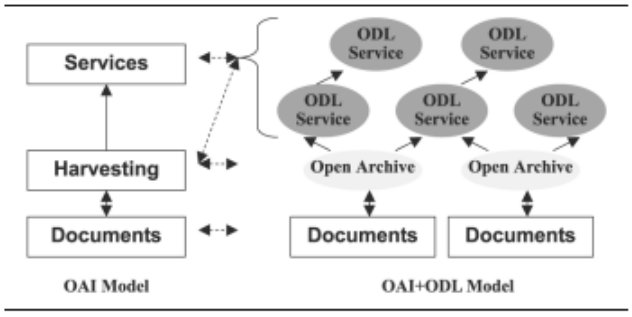


**Figure 3: ODL and OAI architecture diagrams**

### 1.4 Discussion

The three architectures that have been discussed here can be separated into two distinct categories. The first category being a component based architecture, of which the Agent Based Architecture and the Open Digital Library architecture fit into and the second being a more monolithic architecture which the layered architecture fits into.

The agent based architecture does have some similarities to the layered architecture, in the sense that agents are used to fulfill the same functions that the layers fulfil[3]. The main advantage of using an agent based approach over a layered approach is that it allows for greater scalability and flexibility. The main reason for this is because an agent is able to be created in isolation, and easily be integrated into the system without having to change much other logic. In the layered architecture logic in the upper layers will have to be modified to accommodate this new change. One of the problems the agent based architecture suffers from is performance. Agents need to communicate with each other to perform functions, and this communication step adds extra time. Other architectures such as the layered architecture don't suffer from this.

ODL much like the agent based architecture is made up of various components that provide services, however these components are sourced from other digital libraries through the use of XPMH. This architecture has a massive advantage over the other two discussed simply because using this architecture one has access to far more information than the other two architectures simply by design. Unlike the agent based architecture this architecture does not suffer as much from performance problems: Internal user requests are very quick and using the ODL/OAI is faster than spawning other server side processes[4].

Across all the architectures discussed here the Agent Based architecture and the Open Digital Library have the key advantage of being more easily scalable than the Layered Architecture which is a key requirement for digital libraries.

## 2 Repository architecture

The repository is the key component of a digital library since all services or layers will have to interact with it. Hence the organization and storage of data is key. The core requirements a digital repository needs to have are: Long Term Preservation / Access to the repository's content, well organized Meta-data, Interoperability, Security/ User Certification and organization of digital objects[2].

Metadata is one of the most important pieces of information in a digital library and hence how it is stored in a repository is a key consideration when designing a digital repository. There are three approaches to its storage: relational biased solution, Document-biased solution and unbiased solution.

### 2.1 Approach One – a relational biased solution

In this solution Objects and Metadata are stored in a relational multi-media, database. Files are stored as BLOBS and an ER schema will show how to connect these tables in the DB to tables holding meta data.

### 2.2 Approach two – Document biased solution

Files are stored in database(with pointers/labels) and metadata stored in an XML document repository. Business logic is hence needed to associate the two components, requiring a lot of code. In addition unique ids are needed in both xml and actual files.

### 2.3 Approach three – unbiased solution.

Use a relational DB with native XML support. Metadata stored separately from the binary files, in XML documents that follow their own schema. This approach is the most common approach [2].

This approach has several benefits over the above two approaches. With regard to approach (2.2) this approach fixes the problem of needing business logic to link the metadata and the binary data, due to both of them now being managed by one data base management system. This approach is also a direct improvement to approach (2.1) since the metadata is stored separately. This allows for an easy way to contribute to the OAI and allows for metadata about an object to be preserved even if the actual object is removed somehow.

### 2.4 FEDORA

FEDORA is an archiving tool which can be used to create digital repositories. This repository is ideal for storing complex digital objects[5]. This model consists of object models that are templates for various digital objects that are stored in the repository. These object models describe and control what the repository can do with the actual files being stored[6]. This kind of architecture may be useful to consider when designing our archive of archives because an archive itself could be classified by one of these object models.

## 3 Toolkits for building digital libraires

Digital libraries are very complex structures with many components, independent of which architecture you use. In order to simplify the creation of digital libraries, toolkits can be used. The toolkits that will be discussed here are the SimpleDL toolkit, FEDORA toolkit and Dspace

### 3.1 SimpleDL

SimpleDL is a tool for creating pre-generated digital libraries in a low resource environment[7]. SimpleDL allows for the long term preservation of data and retains data through network failures or computer system crashes. In addition, simpleDL allows for the easy migration of data.

This toolkit has several benefits for the archive that we are building. Most notably the fact that it has support for offline functionality, which is something our archive needs to have. A disadvantage of simpleDL is that our archive will store a lot of digital objects and simpleDL can only return results in a feasible amount of time for up to 100 000 items, of which our digital repository may contain more.

### 3.2 FEDORA

Another common tool that specializes in assisting in digital repository design is FEDORA. This service architecture consists of three layers: Web Services Exposure Layer, the Core Subsystem Layer and the Storage Layer[6]. This tool is useful since our digital library needs to be able to store complex objects(other archives). This tools main features include: XML submission and storage, Parameterized disseminators, Access Control and Authentication, Default Disseminator, Searching, OAI Metadata Harvesting and Batch Utility. Features such as versioning, which is an important requirement for our archive, will be made available in future updates. Fedora has four main use cases: Fedora "out-of-the-box", A digital asset management system, A digital library for a research university and Fedora for distributed content objects. In practice a typical implementation uses a blend of all four of these use-cases.

[8] presents a case study of where FEDORA was used to create an extensible digital repository. The digital repository in this paper has a similar goal to ours since this repository was also used to store already made collections of digital objects. The Fedora architecture allowed them to have: support for heterogeneous data types; (2) accommodation of new types as they emerge; (3) aggregation of mixed, possibly distributed, data into complex objects; (4) the ability to specify multiple content disseminations of these objects; and (5) the ability to associate rights management schemes with these disseminations.

### 3.3 Dspace

Dspace is another tool used for creating, normally, institutional digital libraries[9]. This tool provides many core aspects a digital library requires robust repository architecture, search and browse functions, web user interface, ingesting functionality and OAI support, all in one toolkit. Dspace orders members of the community that will use the digital library in a hierarchical fashion to determine their interaction permissions (namely addition and deletion permissions)[16]. Dspace follows a layered architecture approach, consisting of three layers: Application Layer, Business logic layer and storage layer. Dspace is considered to be the most popular tool for creating digital libraries[17].

### 3.4 Discussion

SimpleDL has many differences with Dspace and FEDORA. This toolkit is designed to create libraries that are a lot smaller in scale to libraries created with DSpace and FEDORA. SimpleDL adopts a more lightweight approach to developing digital libraries, so it wont have all the features Dspace and FEDORA have. What SimpleDL does that DSpace and FEDORA cant do, is that it provides offline functionality and hence reduces the dependency on remote servers. SimpleDL is also a lot simpler to maintain and build since there are less moving parts. One notable advantage simpleDL has over FEDORA is that simpleDL comes with user interface and search engine features. If one uses FEDORA these have to be integrated with it[17]. Like Dspace simpleDL also allows the feature for user profiling allowing one to set permissions of users. Of course Dspace and FEDORA have the advantage of doing a lot of what SimpleDL can do but at a much larger scale. Examples being, that libraries made using these tools,

store more data and access data faster than simpleDL libraries and the storage of more complex digital objects can be accommodated.
FEDORA and Dspace are both tools that are used in creation of large institutional digital libraries hence they do share some similarities. The tools are compared in the table below.

| Comparison points | Dspace | FEDORA |
|---|---|---|
| **1. OAI support** | Yes | Yes |
| **2. UI and search feature** | Yes | Needs to be integrated with software that performs that function |
| **3. Architecture** | Layered: Application, Business logic and storage layer | Layered: Webservices, core subsystem and storage layer |
| **4. Storage of digital content** | Stored as items which are made up of Bundles, Bitstreams and Bitstream formats all of which make up different representation of the file | Stored as objects with mappings to different representations of that objects(for example file formats) |
| **5. File formats supported** | Can store any type | Can store any type |
| **6. Complexity** | Used primarily to create institutional wide digital libraries | Can be used to design solutions for simpler problems not just for institutional digital library design. For example it can just be for storage and management of simple content objects(e.g. Jpegs, pdfs) |

**Figure 4: Comparison table comparing important aspects of digital library toolkits between FEDORA and Dspace**

## 4 Offline Archiving approaches

A key requirement for our archive is for it to be operational in an offline environment, in other words, have many of the services that a digital library provides be run on the clients browser, instead of some server. Research has been done into this field and this section will examine in browser search engine designs.

**Approach One**

In this first approach[10] Ajax is used as the tool to create many client side digital library services. One such example is an in-browser query system. All data is indexed and the inverted files as well as the mapping of id to actual name are stored in XML files. Each inverted file contains part of a document id to link documents of its type, allowing for simple queries.

**Approach Two**

The second approach uses an extended Boolean model, and was designed based off of typical information retrieval policies described in Managing Gigabytes[11]. It uses two applications: create index.pl and search.js. search.js simply builds the indices needed by producing lists of inverted indices for each field. Search.js uses that index to locate the item and display it. This search engine has good performance with the most complicated search taking less than half a second to complete for collections of 32000 items[11].

**Approach Three**

Another approach that allows for offline functionality would be to use the Greenstone architecture. This approach involves indexing collections and then distributing them on a CDROM[11]. Users use the service by first selecting a collection they want to use then, use browsing terms to further filter the results and then can use search terms to find individual words or phrases that occur in selected parts of the document[12]. At a high level the system works by organizing the data into collections, each of which have five directories(import, GML, indices of the collection, building information and support files(e.g., configuration files). When new additions to collections are made, importing occurs, in which source material is converted to GML(Greenstone markup language), which includes any metadata that comes with the document. The building step then occurs; which index the data[12].

**Discussion**

All three of these techniques are very suitable to design digital libraries or services of digital libraries. Ajax is a more flexible approach compared to Greenstone since Greenstone requires the installation of their own operating system and Ajax needs just use the technology built into browsers. Ajax technology as mentioned earlier is used to build individual services that can be integrated into digital library systems. This is advantageous over Greenstone since Greenstone is a full package tool and cannot be easily integrated with other client side services.
Searching is an important service in digital libraries, each of which the above approaches are able to perform. These search engines have been evaluated in different ways. The Ajax based

search engine has been used in the Bleek and Lloyd collection[10]. The search engine used in approach two has been evaluated through a number of experiments ultimately theorizing that this approach can be used on collections of 100 000 items feasibly[11]. Greenstone's search service has been long established and is able to operate on collections that have several thousand to millions of records[12]. While it seems Greenstone is the best option to choose, all of its features are not necessary when designing smaller repositories. Approach one and two are hence advantageous to use in the design of smaller repositories.

## 5 Web Archiving

Web archiving is a branch of archiving in which the digital objects stored are other websites. Decisions related to selecting, acquiring, organizing, storing, describing and providing access, are considered when doing this process[13]. This section is going to focus on acquiring(Crawlers: HTTrack, Heritrix and WGet) and storing(WARC format).

### 5.1 HTTrack

HTTrack is a free and open source crawler[13]. One of the main properties of this crawler is that the URL structure of the archived websites is mirrored by the way this crawler writes the websites to disk(recursive). Because of this links no longer take you to the live website but rather to the copy stored on the disk. This may not be a problem for our, archive however another problem about this crawler is that it has slow speeds for crawls and for opening the harvested content.

### 5.2 Heritrix

Heritrix is a widely used crawler that has been used as an alternative to HTTrack[13]. Heritrix stores all harvested web-content in ARC or WARC containers. Another useful feature it has is that it can be configured to ensure that data is not duplicated in the repository if the same website content is received from future crawls. Notable limitations include[14]: There are no scheduled revisits to areas of interest to receive updates to stored objects, Limited ability to recover from in-crawl hardware/system failure, requires sophisticated operator tuning to run large crawls within machine resource limits and it cannot coordinate with other instance of Heritrix performing crawls.

### 5.3 WGet

The final crawler that will be discussed is WGet[15]. WGet is a utility tool for downloading website content, which can be configured as a crawler and downloads files to the depth specified by the user. It uses HTTP, HTTPS, FTP, and FTPS, which is the most widely used Internet Protocols. According to [15] This tool is also a non-interactive command line tool so it can be called from scripts, cron jobs and terminals without X-Windows support. This crawler delivers WARC output. This crawler also is able to continue running when there is no network connection. It wont harvest any data but will continue trying to until the network connection is restored.

### 5.4 ARC and WARC

All of the crawlers discussed are able to assist in the acquisition a huge amount of data from the internet. The ARC and WARC file formats are used as a way to store all this data compactly[18]. The ARC file format has some notable limitations in comparison to the WARC format. The main limitation being that it is unindexed meaning that to access web data one needs to scan the entire ARC file. WARC remedies this problems with certain format enhancements[19]. This allows files stored in WARC to have operations performed on them a lot easier for example migration of data from old formats to newer formats while still linking preserving and linking the old file formats to the newer file formats.

## Summary

This review discussed topics related to the building of digital archives. The first important step to building an archive is to look at the architecture that it will be designed from. The architectures discussed here included: layered architectures, agent based architectures and ODL. The key points to take away here was that the agent based architecture and ODL were more scalable and flexible as opposed to the layered architecture.

Various toolkits for building digital libraries were then examined. Each of these toolkits were compared based on: the complexity of a library that can be built by them; features present in some and not in others and the way they stored and distributed their contents. It was found that Dspace is the best institutional library tool and that SimpleDL is the best choice for smaller library development. Offline or client side services were then examined, in particular searching and browsing, and compared to one another in terms of performance and uses cases. The technologies compared were AJAX, a Boolean model and Greenstone. It was found that both AJAX and Boolean model approaches worked well for smaller library design. The final section spoke about web-archiving in which: three types of crawlers were examined and file formats ARC and WARC, which these crawlers typically store data in. It was found that both WeGet and Heritrix and WARC file formats are better to use at this point in time.

## References

[1] Rhicha Pandey.2003.Digital Library Architecture, Volume 25: DRTC Workshop on digital libraries: Theory and practice

[2] Dimitrios A. Koutsomitropoulos, Anastasia A. Tsakou, Dimitris K. Tsolis, Theodore S. Papatheodorou.2004, Towards the Development of a General-Purpose Digital Repository, Vol: ICEIS (5)

[3] William P. Birmingham. 1995, An agent-based architecture for digital libraries, *D-Lib magazine* 1(July 1995), URL http://www.dlib.org/dlib/July95/07birmingham.html

[4] Hussein Suleman, Edward A Fox, Rohit Kelapure, Aaron Krowne, Ming Luo. 2003 Building digital libraries from simple building blocks. *Online Information Review* (November 2003), 301-310. DOI: 10.1108/14684520310502252

[5] Lagoze, C., Payette, S., Shin, E. and Wilper, C., 2005. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2), pp.124-138. DOI https://link-springer-com.ezproxy.uct.ac.za/content/pdf/10.1007/s00799-005-0130-3.pdf

[6] Staples, T., Wayland, R. and Payette, S., 2003. The Fedora Project. *D-Lib Magazine*, 9(4). DOI https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.9194&rep=rep1&type=pdf

[7] Hussein Suleman. 2021, Simple DL: A toolkit to create simple digital libraries, *International Conference on Asian Digital Libraries*, 2021, Springer, 325-333. https://pubs.cs.uct.ac.za/id/eprint/1512/1/paper_88.pdf

[8] Kumar, A., Saigal, R., Chavez, R. and Schwertner, N., 2004. Architecting an Extensible Digital Repository. In *ACM/IEE CS joint conference on digital libraries*. JCDL. DOI https://dl-acm-org.ezproxy.uct.ac.za/doi/10.1145/996350.99635

[9] Robert Tansley, Mick Bass, David Stuve, Margret Branschofsky, Daniel Chudnov, Greg McClellan, and MacKenzie Smith. 2003. The DSpace institutional digital repository system: current functionality. In Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries (JCDL '03). IEEE Computer Society, USA, 87–97.

[10] Suleman, H. (2007). in-Browser Digital Library Services, in Kovacs, Laszlo, Norbert Fuhr and Carlo Meghini (eds): Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference (ECDL 2007), 16-19 September, Budapest, Hungary, pp.462-465, Springer.

[11] Hussein Suleman (2019) Investigating the effectiveness of client-side search/browse without a network connection, Proceedings of 21st International Conference on Asia-Pacific Digital Libraries (ICADL), 4-7 November 2019, Kuala Lumpur, Malaysia, Springer. Available DOI

[12] Ian H. Witten, Stefan J. Boddie, David Bainbridge, and Rodger J. McNab. 2000. Greenstone: a comprehensive open-source digital library software system. In Proceedings of the fifth ACM conference on Digital libraries (DL '00). Association for Computing Machinery, New York, NY, USA, 113–121. https://doi-org.ezproxy.uct.ac.za/10.1145/336597.336650

[13] Pennock, M., 2013. Web Archiving, *DPC technology watch report*, (2013) 1-45

[14] Mohr, G., Stack, M., Ranitovic, I., Avery, D. and Kimpton, M., 2004. An Introduction to Heritrix, *4th International Web Archiving Workshop, 2004, 109-115*
DOI
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.6877&rep=rep1&type=pdf

[15] Oliveira, C., 2017. *Itsy-Bitsy Spider:* A Look at Web Crawlers and Web Archiving
https://miap.hosting.nyu.edu/program/student_work/2017fall/17f_1807_Oliveira_a2a_y.pdf

[16] Kurtz, M. (2010). Dublin Core, DSpace, and a brief analysis of three university repositories. *Information technology and libraries*, *29*,1, 40-46.

[17] Khan, S. 2019. DSpace or Fedora: Which is a better solution?. *SRELS Journal of information management*, *56*,1, 45-50.

[18] Jaffe, E., & Kirkpatrick, S. 2009. Architecture of the internet archive. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, 1-10

[19] Strodl, S., Beran, P. P., & Rauber, A. 2009. Migrating content in warc files. In *The 9 th International Web Archiving Workshop (IWAW 2009) Corfu, Greece*, 43-49.

.