Chapter 6 Managing cultural heritage: information systems architecture

Lighton Phiri and Hussein Suleman

Department of Computer Science, University of Cape Town, South Africa

1. INTRODUCTION

This chapter is about the architecture of systems that store, preserve and provide access to digital cultural heritage objects. It presents some major design considerations for implementing cultural heritage system architectures and some existing architectural patterns currently in use. Then, a simpler architectural design is proposed; this new architecture could potentially have a positive impact on digital preservation.

Digital Library Systems (DLSes) are specialised Information Systems designed to store, manage and preserve digital content over long periods of time. With the increase in the number of historical artefacts being digitised, the cultural heritage space is one of many application domains where DLSes are currently used, in an effort to foster easy access to this information and additionally preserve the digital content for future use.

While the motivation for using cultural heritage DLSes (hereafter also referred to as cultural heritage systems) is common across systems, the architectural choices made when designing cultural heritage tools and services varies. The variation in the architectural designs are, in part, influenced by the type—video, audio, digital scans, multi-dimensional models etc.—of cultural heritage artefacts that will be subsequently digitally preserved and how the digital objects will be subsequently accessed. Section 2 highlights these requirements further.

The high-level design of these systems takes the form of an architectural framework composed of three main components: a repository layer that stores and manages the digital objects; a service layer with necessary services required to access and manipulate the digital objects; and a user interface layer used by end users to access the digital objects (Arms, 2001). This is illustrated in Figure 1, with specific examples of content and services indicated at each layer.

The remainder of this chapter is organised as follows: Section 2 describes the major resource requirements for cultural heritage systems; Section 3 describes some design constraints and architectural patterns associated with cultural heritage systems; and, finally, Section 4 presents a proposed architectural design aimed at ensuring that the resulting tools and services simplify the overall preservation life-cycle.

2. RESOURCE REQUIREMENTS OF CULTURAL HERITAGE SYSTEMS

The general technological requirements for designing and implementing cultural heritage systems were summarised in the RLG/OCLC report on “Trusted Digital Repository: Attributes and Responsibilities” (Beagrie, Doerr, et al., 2002). This comprehensive list of issues includes: roles and trust; financial issues; organization/legal responsibility; preservation; collections and content; designated communities; and certification. In addressing these issues, organizations with limited resources need to specifically pay attention to the following aspects:

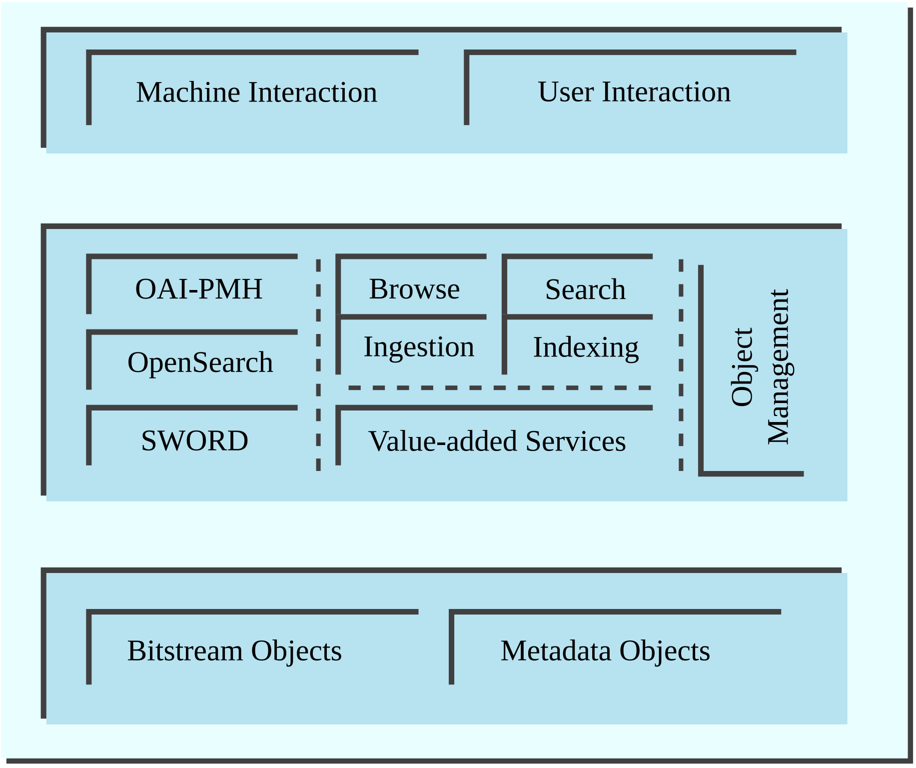

Figure 1. High level architecture of a typical Digital Library System

2.1 Routine maintenance

The preservation life-cycle of digital objects is an on-going process that typically involves the management of digital content and its associated representational information. This routine maintenance is a crucial task in long-term digital preservation for both small- and large-scale preservation projects (Beagrie and Jones, 2001). However, smaller organizations involved in curation and preservation of information often do not have sustainable funding models, making it difficult to effectively manage the preservation life-cycle, as most data and services require regular maintenance.

2.2 Technical expertise

Like other systems, the hardware and software stack used to host cultural heritage systems requires constant and active monitoring. Such activities require specific technical expertise and this effectively raises the management and maintenance costs.

2.3 Technological resources

The vast majority of modern cultural heritage systems are Internet-based and require storage and computational resources, whether local or remotely-hosted. In addition, multimedia applications (such as image/video archives) make intensive use of Internet bandwidth. These requirements may not pose much of a challenge to well-established cultural heritage organizations, but smaller organizations with fewer resources need to plan for this. In addition, when cultural heritage systems are deployed in regions were Internet bandwidth is unreliable and mostly very expensive, it is difficult to guarantee widespread accessibility to services offered.

3. MAJOR DESIGN CONSTRAINTS AND PATTERNS

There are a number of design constraints and patterns associated with cultural heritage systems in particular, and Digital Library (DL) systems in general, including. Some of the key aspects are as follows:

3.1 Scalability

Scalability refers to the ability of a system to expand in order to handle an increasing load. Scalability is an important characteristic for information management systems, and especially DLSes, since there is always a likelihood of adding additional digital objects/collections. It is thus imperative that the architectural design of such systems take into account the potential future growth of content being stored in them.

The scalability of a system can be achieved in two ways: vertical scalability and horizontal scalability. Vertical scalability involves adding resources within an existing logical unit so as to increase capacity. For instance, expanding storage would involve additional hard drives. Horizontal scalability, on the other hand, involves combining multiple logical units of resources to make them function as a single unit (Bondi, 2000).

In the context of cultural heritage systems, the architectural design must be scalable enough to handle: different content types that could potentially be placed in the system; the number of users who will be accessing the system at a given time; and also the different ways through which the content will be accessed.

While the overall decisions made in order to result in scalable architectures might be system-specific, scalability requirements are design-time decisions.

3.2 Preservation

One of the core functions of cultural heritage systems is centred around digital preservation—ensuring that the stored content will be accessible over a long period of time.

A particularly challenging design consideration for cultural heritage systems is ensuring that the architecture is appropriately designed to support potential migration and emulation techniques that might be employed to recover and subsequently access the digital content in the future (Becker et al., 2009). However, this eventual access of digital content is hampered by digital obsolescence—a situation in which software and hardware used to store digital content becomes obsolete due to the rapidly-changing hardware and software environments.

While the effects of digital obsolescence manifest at varying levels of the infrastructure used to store digital content, a cheaper and potentially effective alternative involves using architectures that place an emphasis on ensuring that data formats persist into the future. One example where such an approach has proven viable is that of Project Gutenberg (Hart, 1992). Its success is in part attributed to one of its core principles that all electronic texts are made available in the simplest, easiest to use forms, independent of software and hardware platforms used to access the texts.

3.3 Federated architectures

In a connected society, information sharing is been pivotal in ensuring the success of this connectedness. Federated information system architectures, coupled with interoperability protocols like the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) (C. Lagoze et al., 2002), have been particularly instrumental in facilitating the sharing of information among cultural heritage systems.

Federated architectures function through the seamless integration of distributed independent services that communicate with one another using standardised protocols.

The Europeana project presents a classic example of a large-scale cultural heritage federated system. Europeana is a large-scale cultural heritage portal that offers a single access point to European cultural heritage collections. The portal provides access to millions of digital objects from across Europe via a multilingual interface (Purday, 2009). Being a federated service, resource metadata from disparate data providers and aggregators is periodically loaded into a central database so as to provide a globally consistent view of cultural heritage digital objects for potential end-users of the portal. Dublin Core is used as the resource metadata format, making it easier for standard metadata elements to be put into a Solr search engine (Dekkers et al., 2011).

3.4 Portable architectures

The vast majority of publicly-accessible cultural heritage systems are set up as Web-accessible services. However, some environments do not have the necessary infrastructure required to host such services. In addition, such Internet infrastructure, if available, is expensive.

A number of portable architectures have been proposed to help facilitate universal access to cultural heritage systems. Greenstone[1] is an example of a widely-used software application for heritage systems that are based on a portable architecture. Greenstone is an open-source software tool that was specifically designed for building and distribution of digital collections. The architecture of the tool makes it possible for digital content to be organised and published on the Internet, or optionally on self-installing CD-ROMs (Witten et al., 2001). The software's ability to redistribute collections on a self-installing CD-ROM has made it a popular tool in regions with limited Internet connectivity.

The digital Bleek and Lloyd collection[2] is another example of a cultural heritage system implemented using a portable architecture. It is an online catalogue that was developed to store and enable access to digitised manuscripts depicting the life of the |Xam and !Kun speakers of Southern Africa. The system was designed to be XML-centric, and is based on an implementation strategy that involves pre-generating scalable hyperlinked XHTML pages using XSLT (Phiri and Suleman, 2012; Suleman, 2007).

4. DESIGNING FOR PRESERVATION

4.1 Motivation

Digital Libraries were initially designed, to meet the above objectives and address the complex identified issues, as an abstraction layered over databases to provide higher level services (Arms et al., 1997; Baldonado et al., 1997; Frew et al., 1998). They have subsequently become more complex (Janée and Frew, 2002; Carl Lagoze et al., 2006), and thus difficult to maintain, extend and reuse. The difficulties resulting from the complexities of such tools are especially prominent in organizations and institutions that have limited resources to manage such tools and services. Some examples of organizations that fall within this category include cultural heritage organizations and a significant number of other organizations in developing countries found in regions such as Africa (Suleman, 2008).

The majority of existing tools for cultural heritage curation are arguably unsuitable for resource-constrained environments due to the reasons outlined in Section 2. Thus, an alternative architectural design was pursued, based on a set of defined design principles.

---

[1] www.greenstone.org

[2] http://lloydbleekcollection.cs.uct.ac.za

4.2 Principles

A grounded theory qualitative analysis of successful architectures for DLSes, and other systems to manage digital content, was conducted. The fundamental outcome of this analysis, and interaction with organizations needing to preserve digital content, is a set of guiding design principles, as described in the following sections.

4.2.1 Principle 1: Hardware and/or software platform independence

It should be possible to operate tools and services on a wide variety of hardware and software platforms. The rationale behind this principle is to ensure that the least possible cost associated with technological infrastructure is incurred during the collection management life-cycle.

The preservation life-cycle of digital objects is an on-going process that typically involves the management of digital content and its associated representational information. The cost implications of long-term digital preservation is a crucial task for both small- and large-scale preservation projects (Beagrie, Doerr, et al., 2002). However, the vast majority of organizations involved in the curation and preservation of digital information usually do not have adequate funding to support this process, which often includes migration of tools and/or content.

A reduction in the cost associated with the collection management process could be achieved in various ways including, but not limited to, the following:

- Designing tools that require minimal technical expertise to manage
- Designing tools capable of being run on popular operating systems
- Designing tools capable of being operated on hardware platforms with minimal specifications

4.2.2 Principle 2: Heterogeneous object, metadata and service integration

There should be explicit support for integration of any digital object type, metadata format or new service.

The proliferation of both born-digital and digitised information has given rise to various data formats and metadata standards. In addition, there is a growing demand for DL services in order to facilitate ubiquitous access to information.

It is therefore necessary that the design of DL tools be flexible enough to accommodate heterogeneous objects, metadata and services.

4.2.3 Principle 3: Support for community and international standards

The design of tools and services should take into account community-based standards and international standards in order to facilitate interoperability.

The increase in the amount of digital content generated and made available publicly has brought about a need to standardise processes in the digital curation workflow. Incorporating standards in the initial stages of the design process would effectively ensure that the resulting DL services become interoperable with other external services. It also makes it easier for services to be customised.

4.2.4 Principle 4: Flexible design to facilitate extensibility

The design should be flexible enough to enable end users to adapt the tools and services to their own needs.

DLSes are increasingly being used in a wide array of application domains, e.g., institutional repositories, cultural heritage systems. The services offered by these different application domains vary and the overall design must be flexible enough to facilitate customisation and extensibility.

4.2.5 Principle 5: Minimalist design approach

There should be minimal use of external software components in order to simplify the overall design. This would arguably result in tools that are easier to manage.

The design of services should, at a minimum, only be composed of the least number of components that are required for it to function. Auxiliary external components should be made optional, making them available only when required.

In addition, mandatory components should be critically analysed to ensure that they make use of simplest possible solutions and/or technologies.

4.2.6 Principle 6: Simplified preservation process

The preservation process should be simplified as much as possible to make it possible to easily migrate digital content.

The preservation life-cylce is an on-going process that requires dedicated staff. The majority of contemporary DL services require technology experts to perform the routine preservation tasks.

The overall design should thus be made as simple as possible so that relatively novice users are able to perform at least basic preservation tasks.

4.2.7 Principle 7: Structured organization of data

There should be explicit support for hierarchical logical organization of information.

The majority of data that is curated and made publicly accessible has some form of logical organization of information. In addition, data consumers usually visualize information using varying logical views.

The design should thus explicitly support the logical organization of information, and make it flexible enough for users and/or administrators to define the desired logical views and structures.

4.2.8 Principle 8: Design for least possible resources

There should be support for access to digital collections in environments with resource constraints.

One of the major motivating factors for advocating for a minimalist approach to the overall design of DL tools and services is apparent unavailability of DL tools that can effectively operate in resource-constrained environments. This is still an issue for most environments in developing countries, such as those found in Africa. The design of DL services should thus be based on the least possible resources to enable resulting services to operate in environments with limited resources.

4.3 Repository design overview

The principled design approach, resulting from the principles described in Section 4.2, is applicable to the different architectural components of DLSes —user interface layer, service layer and repository sub-layer. The emphasis in this discussion is on the repository layer.

4.3.1 Design decisions

The repository design decisions are a result of a direct mapping of the principles in Section 4.2 and requirements of the different components of a typical DLS repository sub-layer (Arms et al., 1997).

These design decisions are presented in a series of tables. Table 1 presents the storage design of digital objects. Table 2 presents the storage design of metadata objects. Table 3 presents the scheme for object and metadata naming. Finally, Table 4 presents the object structuring scheme.

Table 1: Simple repository persistent object store design decision

Element	**Description**
Requirement	Pesistent object storage
Issues	Principles 1, 2, 6 and 8
Decision	Store bitstreams on the local operating system filesystem
Assumptions	None
Alternatives	Store bitstreams as blobs in a database; store bitstreams in the cloud
Rationale	Backup and migration tasks associate to repository objects can be potentially simplified; operating system commands can be used to perform repository management tasks
Implications	None –most conventional tools and services use the same approach
Notes	None

Table 2: Simple repository metadata storage design decision

Element	**Description**
Requirement	Metadata records storage
Issues	Principles 1, 2, 5, 6 and 8
Decision	Native operating system filesystem used for metadata storage
Assumptions	None
Alternatives	Relational database; NoSQL database; embed metadata into digital objects
Rationale	Storing metadata records in plain text files ensures platform independence; complexities introduced by alternative third-party storage solution avoided through the use of native filesystem
Implications	No standard method for data access (e.g. SQL); Transaction process support only available via simple locking; non-availability of complex security mechanisms
Notes	None

Table 3: Simple repository object naming scheme design decision

Element	**Description**
Requirement	Object naming scheme
Issues	Principle 5
Decision	Use actual object name as unique identifier
Assumptions	Native operating systems
Alternatives	File hash values; automatically generated identifiers
Rationale	Native operating systems ensure file naming uniqueness at directory level. In addition, it is a relatively simpler way of uniquely identifying objects as object naming control is given to end users, rather than imposing it on them
Implications	Object integrity has a potential to be compromised; objects could potentially be duplicated by simply renaming them
Notes	None

Table 4: Simple repository object storage structure design decision

Element	**Description**
Requirement	Object storage structure
Issues	Principles 6 and 7
Decision	Store bitstreams alongside metadata records –at the same directory level on the filesystem; filesystem directory to be used as container structures for repository objects
Assumptions	The other sub-layers of the DLS have read, write and execute access to the repository root node
Alternatives	Separate storage locations for bitstreams and metadata records
Rationale	Storing bitstreams and corresponding metadata records alongside each other could ultimately make potential migration processes easier; container structures could potentially make it easier to move repository objects across different platforms
Implications	None
Notes	None

4.3.2 Repository architecture

The architectural design is centred around designing a simple repository. At a bare minimum this should be capable of facilitating the core features of a DLS, such as long term preservation and ease of access to digital objects.

The repository design is file-based and makes use of a typical native operating system filesystem as the core infrastructure. Table 5 shows the main components that make up the repository sub-layer, with all the components residing on the filesystem, arranged and organised as normal operating system files—regular files and/or directories—as shown in Figure 2.

Table 5: Repository component structural composition

Component	**File Type**	**Description**
Container Object	Native OS directory	Structure used to store digital objects
Content Object	Native OS regular file	Content/bitstreams to be stored in the repository
Metadata Object	Native OS regular file	XML-encoded plain text file for storing metadata records

A typical DLS repository would be located in an application-accessible base root directory node, and is composed of two types of digital objects—Container Objects and Content Objects—both of which are created and stored within the repository with companion Metadata Objects that store representational information associated with the objects. Figure 2 illustrates how Container and Content objects are stored on a typical filesystem.

Container Objects can be recursively created within the root node as the repository scales, and enable the creation of additional Container Objects within them. As shown in Figure 3, the Metadata Object associated with a Container Object holds information that uniquely identifies the object; optionally describes the object in more detail, including relationships that might exist with other objects within the repository; and a detailed log of objects contained within it, referred to as the manifest.

Content Objects represent digital objects to be stored within the repository. As shown in Figure 4,

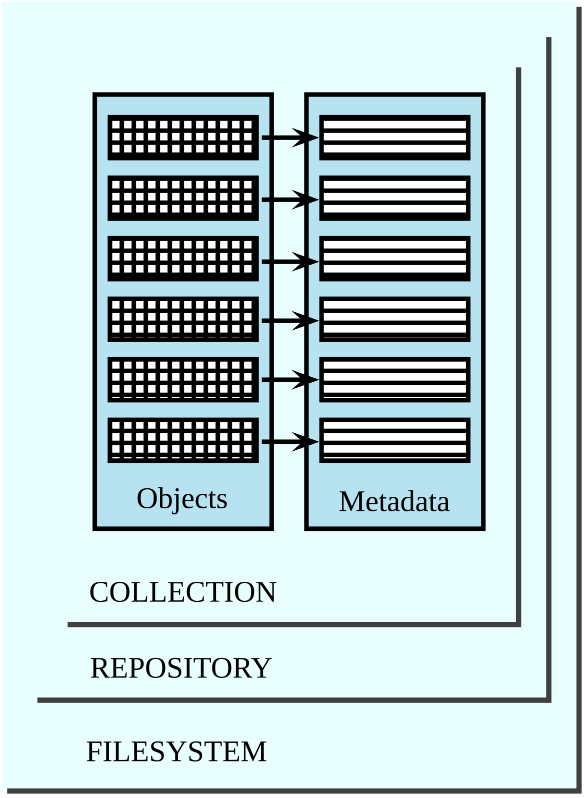

Figure 2. Repository object organization

the representational information stored in the Metadata Objects associated with Content Objects is similar to that of Container Objects, with the exception of manifest-related information.

4.4 Case studies

This repository architecture was implemented in two case studies, described below, to assess feasibility and examine the impact of these design decision on real-world data.

4.4.1 Case study: The Bleek and Lloyd collection

The Bleek and Lloyd collection (Skotnes, 2007) is a 19th century compilation of notebooks and drawings comprising of linguistic and ethnographic work of Lucy Lloyd and Wilhelm Bleek on the life of the |Xam and !Kun speakers of Southern Africa. In 2003, the Lucy Lloyd Archive and Research Centre at the University of Cape Town embarked on a large scale digitisation project of all the artefacts and corresponding representation information was generated. Table 6 shows the current composition of the digitised objects and Figure 5 shows a sample page from one of the digitised notebooks.

A repository for this collection was implemented using the hierarchical architectural design described in Section 4.3.2. The container and digital content metadata records are encoded using qualified Dublin Core for descriptive metadata tags and relationships that exist within the different resources.

A prototype Web-based DLS, that makes use of this repository as the data storage layer, was subsequently implemented (Phiri, Williams, et al., 2012) using the Java programming language. The prototype DLS is made up of two main components: a curator interface for performing digital object

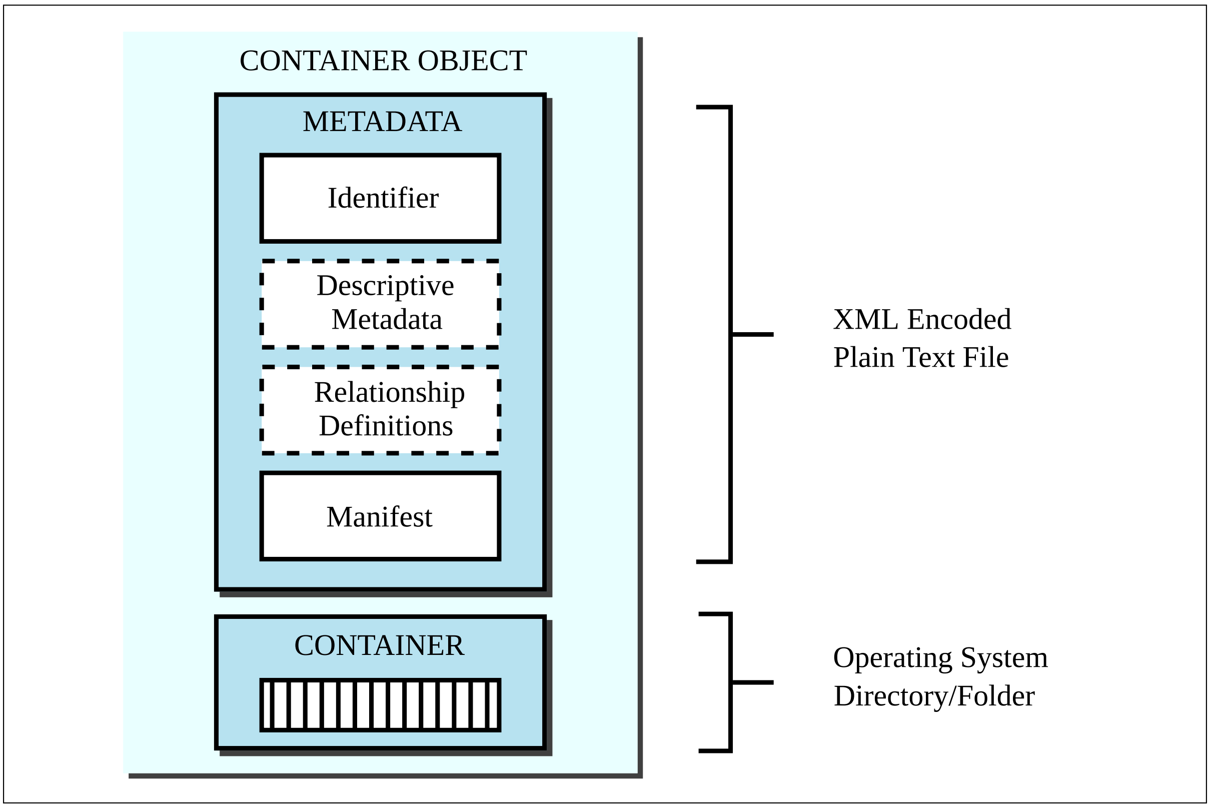

Figure 3. Simple repository container object component structure

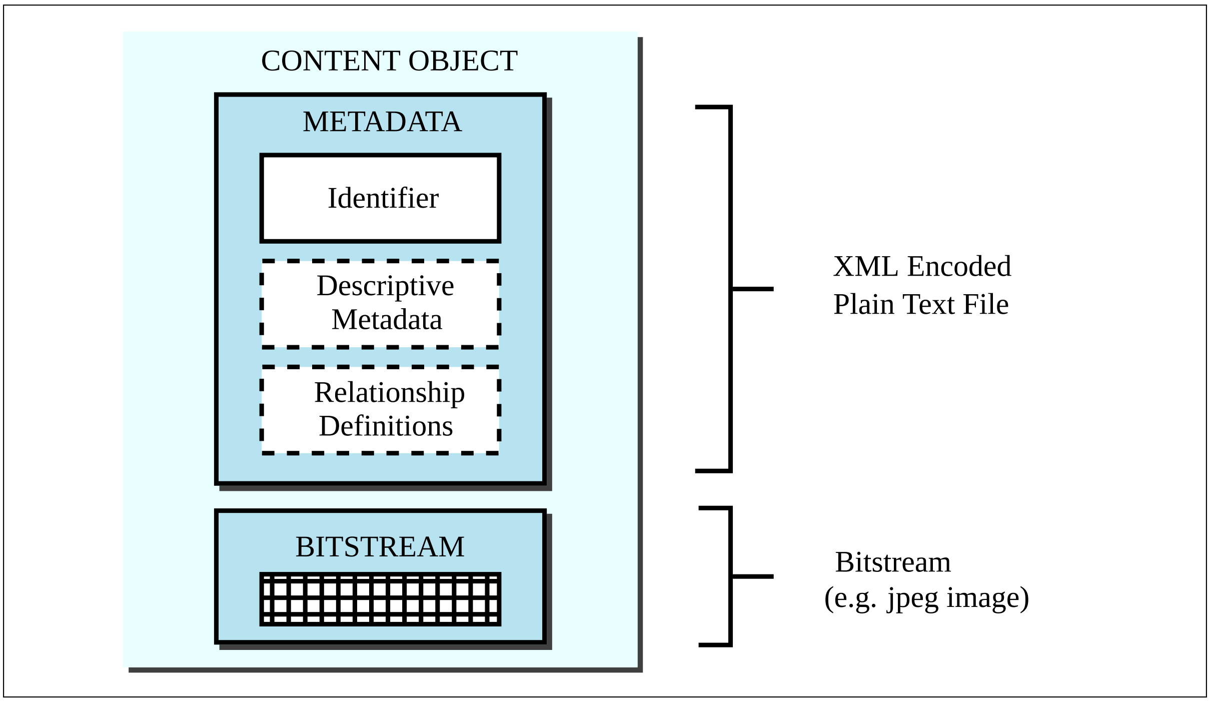

Figure 4. Simple repository content object component structure

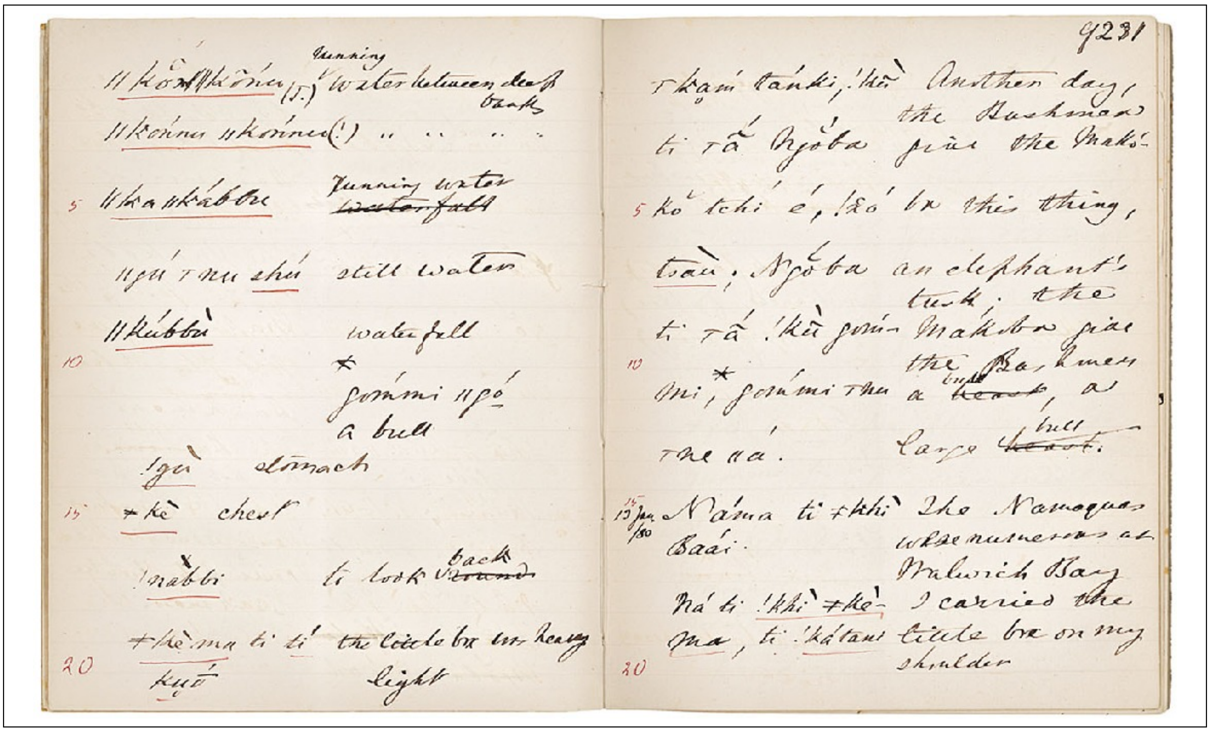

Figure 5. Screenshot showing a sample page from the “Posts and trading” story in the Lucy Lloyd !Kun notebooks

management tasks and administrative functions; and an end-user interface, through which end-users can access the digital content stored in the repository. Evaluation of these interfaces confirmed that their usability, utility and performance were acceptable. There was no discernible impact on user experience.

Table 6: Bleek& Lloyd collection profile

Collection theme	Historical artifacts; museum objects
Media types	Digitised
Collection size	6.2GB
Content type	image/jpeg
Number of collections	6
Number of objects	18 924

4.4.2 Case study: SARU archaeological database

The Spatial Archaeology Research Unit (SARU), in the Department of Archaeology[3], at the University of Cape Town has been compiling archaeological collections since the early 1950s. These collections are predominantly in the form of site records and corresponding artefacts within the vicinity of the sites. Table 7 show the composition of collections that have been compiled thus far, and Figure 6 shows an image of a rock art motif from one of the archaeological sites.

Owing to the growing number of collections and a growing need by a number of researchers to access this information, an archaeological database was designed in 2005, in part, to produce layers suitable

---

[3] http://web.uct.ac.za/depts/age

for integration with Geographic Information Systems. The site records are originally accessible only via a Microsoft Access[4] database-based desktop application (Wiltshire, 2011).

Using the repository architectural design described in Section 4.3.2, a hierarchical file-based repository was implemented to store artefacts for the SARU collection. However, unlike the Bleek and Lloyd collection outlined in Section 4.4.1, a custom metadata scheme for the digital content had to be devised due to the complex nature of the metadata records associated with the artefacts.

The School of Rock Art cultural heritage educational portal (Crawford et al., 2012) was implemented as a layered service on top of the file-based repository. The portal is composed of three main components: a Cave Navigation module to enable end users to navigate three dimensional models of caves, annotated with repository images; a Guided Tours module for sequencing the viewing of repository images; and a Story Telling module that integrated stories with images from the repository. This case study demonstrated usability, flexibility and extensibility of the core repository design.

5. SUMMARY

Table 7: SARU archaeological database collection profile

Collection theme	Archaeology artifacts; museum objects
Media types	Born digital
Collection size	283GB
Content type	image/jpeg; image/tiff
Number of collections	110
Number of objects	72 333

Architectures for digital library systems to preserve cultural heritage artefacts have much in common with other forms of DLSes, but also some requirements that are different. In designing such cultural heritage systems, system architects need to focus on the needs of curators as well as the ever-present preservation imperative. Such a preservation-focused effort leads to specific design goals that are arguably well-served by an architectural model based on the concept of simplicity.

This chapter has presented the motivation and details behind such a simple architecture, based on a set of design principles that were derived from an analysis of successful aspects of existing architectures. This repository architecture was then used as the basis for two real-world case studies, which suggest that it is a feasible model. Various user studies and performance experiments have been conducted to prove the flexibility and scalability of such simple architectures, in the context of these case studies and others.

Ultimately, there is no silver bullet in DL system architecture. However, simple architectures may have advantages for some types of systems; and some elements of simple architectures may have wide applicability for all systems. It is clear, however, that the architecture of systems needs to be planned and based on specific goals, as that can have a profound impact of the architectural design and therefore the content being curated and its long-term preservation and access.

---

[4] http://office.microsoft.com/en-us/access

Figure 6. Screenshot showing the Die Mond South plant fossil from the Eastern Cederberg rock art site

BIBLIOGRAPHY

[1] William Y. Arms. “Libraries, Technology, and People”. In: *Digital Libraries*. Ed. by William Y. Arms. 2nd. Cambridge, Massachusetts: The MIT Press, 2001. Chap. Chapter 1, pp. 1–20.

[2] William Y. Arms, Christophe Blanchi, and Edward A. Overly. “An Architecture for Information in Digital Libraries”. In: *D-Lib Magazine. The Magazine of Digital Library Research* 3.2 (Feb. 1997). DOI: 10.1045/february97-arms.

[3] Michelle Baldonado, Chen-Chuan K. Chang, Luis Gravano, and Andreas Paepcke. “The Stanford Digital Library metadata architecture”. In: *International Journal on Digital Libraries* 1.2 (1997), pp. 108–121. DOI: 10.1007/s007990050008.

[4] Neil Beagrie, Marianne Doerr, Bayerische Staatsbibliothek, Margaret Hedstrom, Maggie Jones, Anne Kenney, Catherine Lupovici, Kelly Russell, Colin Webb, Deborah Woodyard, Robin Dale, and Meg Bellinger. *Trusted Digital Repositories: Attributes and Responsibilities. An RLG-OCLC Report*. Mountain View, CA: Research Libraries Group, 2002. URL: http://www.oclc.org/resources/research/activities/trustedrep/repositories.pdf (visited on Sept. 13, 2012).

[5] Neil Beagrie and Maggie Jones. *Preservation Management of Digital Materials: a Handbook*. Ed. by Null The Digital Preservation Coalition. November. British Library, 2001, pp. 31–47. URL: http://discovery.ucl.ac.uk/12228/.

[6] Christoph Becker, Hannes Kulovits, Mark Guttenbrunner, Stephan Strodl, Andreas Rauber, and Hans Hofman. “Systematic planning for digital preservation: evaluating potential strategies and building preservation plans”. In: *International Journal on Digital Libraries* 10.4 (Dec. 2009),

pp. 133–157. DOI: 10.1007/s00799-009-0057-1. URL: http://link.springer.com/10.1007/s00799-009-0057-1.

[7] André B. Bondi. “Characteristics of scalability and their impact on performance”. In: *Proceedings of the second international workshop on Software and performance - WOSP ’00*. New York, New York, USA: ACM Press, 2000, pp. 195–203. DOI: 10.1145/350391.350432. URL: http://portal.acm.org/citation.cfm?doid=350391.350432.

[8] Kaitlyn Crawford, Marco Lawrence, and Joanne Marston. *School of Rock Art*. University of Cape Town, 2012. URL: http://pubs.cs.uct.ac.za/honsproj/cgi-bin/view/2012/crawford_lawrence_marston.zip/index.html (visited on Feb. 14, 2012).

[9] Makx Dekkers, Stefan Gradmann, and Jan Molendijk. *D3.4 Final Technical & Logical Architecture and future work recommendations*. Tech. rep. 2011, pp. 1–29.

[10] James Frew, Michael Freeston, Nathan Freitas, Linda Hill, Greg Janee, Kevin Lovette, Robert Nideffer, Terence Smith, and Qi Zheng. “The Alexandria Digital Library Architecture”. In: *Research and Advanced Technology for Digital Libraries*. Springer Berlin Heidelberg, 1998, pp. 61–73. DOI: 10.1007/3-540-49653-X_5.

[11] Michael Hart. *Project Gutenberg. The History and Philosophy of Project Gutenberg*. 1992. URL: http://www.gutenberg.org/wiki/Gutenberg:The_History_and_Philosophy_of_Project_Gutenberg_by_Michael_Hart (visited on Jan. 9, 2012).

[12] Greg Janée and James Frew. “The ADEPT digital library architecture”. In: *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*. JCDL ’02. New York, NY, USA: ACM Press, 2002, pp. 342–350. DOI: 10.1145/544220.544306.

[13] C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. “Open Archives Initiative-Protocol for Metadata Harvesting-v. 2.0”. 2002. URL: http://www.openarchives.org/OAI/openarchivesprotocol.html.

[14] Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper. “Fedora: an architecture for complex objects and their relationships”. In: *International Journal on Digital Libries* 6.2 (Apr. 2006), pp. 124–138. DOI: 10.1007/s00799-005-0130-3.

[15] Lighton Phiri and Hussein Suleman. “In Search of Simplicity: Redesigning the Digital Bleek and Lloyd”. In: *DESIDOC Journal of Library & Information Technology* 32.4 (July 2012), pp. 306–312. URL: http://publications.drdo.gov.in/ojs/index.php/djlit/article/view/2524 (visited on Oct. 25, 2012).

[16] Lighton Phiri, Kyle Williams, Miles Robinson, Stuart Hammar, and Hussein Suleman. “Bonolo: A General Digital Library System for File-Based Collections”. In: *Proceedings of the 14th International Conference on Asia-Pacific Digital Libraries*. Ed. by Hsin-Hsi Chen and Gobinda Chowdhury. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2012, pp. 49–58. DOI: 10.1007/978-3-642-34752-8_6.

[17] Jon Purday. “Think culture: Europeana.eu from concept to construction”. In: *The Electronic Library* 27.6 (2009), pp. 919–937. DOI: 10.1108/02640470911004039. URL: http://www.emeraldinsight.com/10.1108/02640470911004039.

[18] Pippa Skotnes. *Claim to the Country - The Archive of Wilhelm Bleek and Lucy Lloyd*. Johannesburg, South Africa: Jacana Media, 2007.

[19] Hussein Suleman. “An African Perspective on Digital Preservation”. In: *Proceedings of the International Workshop on Digital Preservation of Heritage and Research Issues in Archiving and Retrieval*. Kolkata, India, 2008.

[20] Hussein Suleman. “Digital Libraries Without Databases: The Bleek and Lloyd Collection”. In: *Proceedings of the 11th European Conference on Research and Advanced Technology for Digital Libraries*. Ed. by László Kovács, Norbert Fuhr, and Carlo Meghini. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 392–403. DOI: 10.1007/978-3-540-74851-9_33.

[21] Nicolas Wiltshire. *Spatial analysis of archaeological sites in the Western Cape using an integrated digital archive*. University of Cape Town, 2011. URL: http://uctscholar.uct.ac.za/R/N8XKKNNCY76DM8GQG33X3C8LCDJ7N6MUKFFKAHMT67HVDJN9TT-03715?func=results-brief (visited on Feb. 14, 2012).

[22] Ian H. Witten, David Bainbridge, and Stefan J. Boddie. “Greenstone: open-source digital library software with end-user collection building”. In: *Online Information Review* 25.5 (2001), pp. 288–298. DOI: 10.1108/14684520110410490.