Latin American and Caribbean Center on Health Sciences Information

Pan American Health Organization | World Health Organization

Project Report

ISIS - Network Based Platform: Final Report

Version 1.0

Sao Paulo - July 2007

ISIS - Network Based Platform: Final Report

Card catalog

BIREME / PAHO / WHO (Brazil)

ISIS - Network Based Platform: Final Report. / BIREME / PAHO / WHO. Sao Paulo : BIREME / PAHO / WHO, July 2007.

49 p.

1. User manual. 2. Information access. 3. Information systems. 4. Data management. 5. ISIS. I. BIREME II. Title

Warning - Any mention in this document to companies, institutions, persons or products are not an endorsement or recommendation given by BIREME / PAHO / WHO, thus it does not mean a preference to a similar one, cited or not.

BIREME / PAHO / WHO

Latin American and Caribbean Center on Health Sciences Information

Rua Botucatu 862 V Clementino

This document was produced with the Documents Conformation Methodology (NorDoc) developed by BIREME.

Table of contents

Project specification by Rodrigo Senra (GPr Sistemas Ltda.) under contract with BIREME/PAHO/WHO under the supervision of a focus group with Milton Lapido, Abel L. Packer, Adalberto Tardelli and Ernesto Spinak as members

CHRONOGRAM AND PROGRESS

ISIS - FLOSS	month 01	month 02	month 03	month 04	month 05
Specification					
Technique Specification					
Surveys of: writing format of the files of the current ISIS					
Surveys of: actual ISIS abilities					
Surveys of: processes carried out by the Current ISIS					
Surveys of: abilities of the format language of the current ISIS					
Compile the set of abilities of the New ISIS					
Validation of the set of ability of the New ISIS					
Preparation of the Technical Specification Document					
Architecture and design					
Survey of the characteristics of the platforms intended for the New ISIS (Zope, JEE)					
Compilation of the characteristics intended in each platform					
Formulation of basic architecture (independent of platform)					
Formulation of the final architecture in each platform					
Preparation of the Document of Architecture and Design for all the platforms					
Quality Control					
Define points for unit test					
Define processes of validation of the functions of the New ISIS					
Define processes for development, test, and distribution of the New ISIS					
Plan					
Development Plan					
Definition and implementation of a communication point among the developers					
Definition of development internal team and management responsible					
Dissemination of the project with the international developer community.					
Presentation of the project documentation to international community of developers					
Development and publication of the basic module					
Implementation of the collaborative development processes					
Project Fases					
Basic storage module					
Import and export module of the current version of the ISIS					
Keyword indexing module					
Indexing module of full text					
Coupling module for Zope					
Coupling module for JEE					
Interpretation module for format language					
XML treatment module in semi-native way					

Executed task Pending task Reprogrammed task

All activities of the Specification part of the chronogram were executed with success and as result we have the final version of the Technical Specification document. The activities in the Plan portion of the chronogram had to be redefined to match the requirements of the final Technical Specification and are described in the Appendix II – Development Plan.

TECHNICAL SPECIFICATION

1 Introduction

ISIS[1] comprises a family of software systems oriented to storage and retrieval of textual based information sources of the bibliographic type. Capable of operating an extraordinary range of applications with different levels of complexity in restrict platforms, ISIS is used worldwide to manage libraries and information systems in different regions, languages and cultures. In addition to library catalogues, ISIS presents efficient solutions to operate full texts and directories, with high flexibility to generate and operate retrieval indexes.

Acknowledging the historical importance of, as well as the persisting demand for its updating, on May 15, 2006, at UNESCO[2] headquarters in Paris, a group of specialists in textual information systems of the ISIS software family recommended the development a new solution in free software. The meeting was attended by representatives from several institutions that use ISIS in their information management platforms. The main goals of the meeting were to present and discuss a proposal prepared by BIREME to develop a new open access network-based ISIS Project, and to set a partnership strategy. Most institutions

1 Integrated Set of Information System
2 United Nations Educational, Scientific and Cultural Organization

represented in the meeting took a standing of being facilitators and willing to contribute with resources, including financial ones.

The ISIS – Network Based Platform Project aims to rethink the architecture and tools of classic ISIS textual databases, aiming to bring that technology to the state-of-the-art in Computer Science and also to adopt agile methodologies and tools following the Free and Open Source Software (FOSS) development model and guidelines.

The classic ISIS model (and software tools) had several incarnations targeting different operating platforms. As technology advanced, ISIS tools adapted to changes and evolved. However, a major restructure of ISIS architecture can ease the path to incorporate new functionality and services, improve performance and interoperability, reduce maintenance costs, stretch scalability and leverage its adoption even further.

Regarding the development methodology, the FOSS model is mature and widely accepted. Furthermore, it fosters socio-economic development and people empowerment in developing countries or regions. It is therefore a natural choice for this endeavour.

This report will set the main objectives of the ISIS-NBP project, explain the motivation behind each objective, present the ISIS-NBP design and architecture, and suggest an implementation plan supported by currently available technological options.

2 Objectives

The ISIS-NBP project has a set of bold, yet feasible, goals:

- overcome critical limitations in capacity and expressiveness of classic ISIS storage data format
- design a scalable digital library architecture, that meets needs ranging from small users to large data providers. This effort should strive to keep a shared code base, in order to reduce maintenance costs and increase robustness.
- ease the coupling between digital libraries and external software systems, increasing interoperability
- create a cross-platform set of tools to ease the creation of end-user applications.
- provide a incremental and evolutionary migration path from classic ISIS databases and applications to become adherent to the model and tools offered by ISIS-NBP.

This report will revisit each of these objectives, explaining the motivations that lead to each goal, proposing solutions and exposing their advantages and disadvantages. Finally, we present action path.

3 Motivations

This section will examine in greater detail the motivation behind each objective. We will present some use cases born from real world practice, followed by a suggested course of action to achieve each goal.

3.1 Rethinking the Model

From the classic ISIS, there are two critical points that need addressing and that are derived from the model: expressiveness and capacity. We will discuss each separately, but prior to that we will conceptualize the CDS-ISIS database model, structure and its storage format.

Summarized from UNESCO's 1989 CDS-ISIS Reference Manual, here is a high-level definition:

> The CDS-ISIS database allows you to build and manage structured non-numerical data bases. Data elements are stored in fields, each of which is assigned a numeric tag (name of the field) indicative of its contents. The collection of fields containing all data elements of a given unit of information is called a record. CDS/ISIS is specifically designed to handle fields of variable length, that may be optional, may be repeatable, and may contain multiple variable length data elements (sub-fields).

This definition exposes an *almost flat* representational model for data. Each information unit is represented by the *field* concept, which has **only one level of nesting** (sub-fields). Moreover, information units (*fields*) can be grouped by to constructs: *records* and repetition. None of these allows arbitrary nesting. Therefore, this model is not suitable to represent hierarchies when there are more than 2 nested levels.

Furthermore, there is no construct to distinguish between core data and metadata, while preserving the locality principle[3]. In a given record, all fields belonging to the record's schema have equal status. There are two obvious solutions to represent core data and metadata in classical CDS-ISIS. The first one is to distinguish data from metadata by an **implicit** tag range **convention.** The second solution would be to store metadata in a **different** record from its core data, binding both records with a reference field. Both approaches are feasible and adopted in practice. The problem with the first solution is that such implicit convention is neither intuitive nor robust. The problem with the second solution is that splitting up core data from its associated metadata is potentially harmful for performance and consistency.

These two issues become critical if we consider the increasing demand to interoperate digital libraries with XML-based data providers and consumers. The XML data model supports arbitrary hierarchical data trees, offers constructs to **explicitly** separate core data from metadata (mark-up and attributes) while **preserving locality of reference.** Therefore, to accommodate the XML model we ought to rethink ISIS *quasi-flat* data model. One immediate consequence would be to **adapt** the ISIS **data definition and query language** to be adherent and complaint with the new extended model.

Another problem in expressiveness is the choice of numerical tags to designate fields. Integer tags do not make the field identifiers self-describing and have a poor mnemonical appeal when interfacing with human operators, in opposition to software or hardware interaction. The same argument is valid for the choice of single characters to designate sub-field identifiers. In both cases, the use of word

---

3 Logically related data should be stored close to each other, aiming to optimise data retrieval.

or phrase identifiers is much more mnemonic and intuitive, and therefore desirable. With all the advances in software and hardware in the last decades, we should focus in providing a better user experience, instead of making systems simply closer to how the machines work, and easier to implement as an excuse for providing poor user experience. Once again, these issues push towards a XML-based model.

Last but not least, the classical CDS-ISIS model had devices focusing on textual-based interfaces such as mainframe terminals and console interaction. One obvious example is the predefined view modes (*proof*, *heading* and *data) and* all the concerns with line breaking and paragraph formatting mark-up. Nowadays, the web is the most widespread media and the browser its natural and ubiquitous interface. In the web realm, text layout is done very differently from the ancient times of the mainframe terminals. Therefore, ISIS traditional formatting language constructs should be adapted to this new medium.

3.2 Rethinking the Storage

In addition to the expressiveness restrictions we have just discussed, there is a major concern about expanding storage capacity. Most of the capacity limitations stem from the chosen binary layout of the data format. The decision to adopt a XML-based repository [2, 7, 8, 9, 10, 26] model will deeply affect the storage layer.

Any binary storage format has some implicit tradeoffs embedded in its layout. Furthermore, these tradeoffs are technology-dependent, and as technology evolves these tradeoffs must be reviewed. The challenge is to adjust the binary layout in order to expand capacity while keeping compatibility with legacy code.

Just to illustrate this issue, the original maximum file size in CDS-ISIS was 512 MB. That limit was extended to 2 GB, which is four times bigger than the original limit. Even this limit is surpassed by modern operating systems. For instance:

- a FAT32 file system, typical in Windows 9x desktops, could hold a 4 GB file.
- a NTFS file system, typical in Windows XP, NT and Vista, could hold a 2 TB[4] file.
- several file systems for Linux/Unix can handle up to 8 TB files.

There are file system supporting even files with a maximum size of 8 EB[5], given proper hardware and software customization.

However, we are not advocating the storage of digital library collection in single or few huge files. The bottom line is that the current ISIS capacity restrictions should not only be reviewed, but we ought to elaborate a design that eases incremental capacity extension to keep up with the pace of storage technology evolution.

Obviously, we are not considering just maximum file size restrictions. There is a wide plethora of attributes that should be considered. To illustrate we name a few key capacity-related attributes:

- maximum record size
- maximum number of records per database
- maximum number of fields
- maximum field size
- number of subfields

File recoverability is another issue related to the storage file format where there is room for improvement. The classical ISIS master file binary layout, when physically corrupted, might be left in a state from which is very hard to identify record boundaries. If the cross-reference file is intact, it could be used successfully to recover the master file, since it encapsulates record boundary information. Nevertheless, the master file and cross-reference file are usually stored contiguous; therefore there is a high risk that they might suffer corruption simultaneously.

On last issue regarding storage restructure is to consider data compression opportunities. Even though the cost of storage devices is decreasing in time, many ISIS users do not upgrade their hardware frequently and have to deal with limited storage capacity. Moreover, the introduction of data compression may be a

---

4 1TB (terabyte) = 1024 GB (gigabytes)
5 1EB (exabyte) = 1048576 TB (terabytes)

strategy to counter-balance a potential database volume growth deriving from the other measures discussed in this section.

3.3 Improving Scalability

One scalability issue is storage capacity that we have dealt with in the previous section. Another scalability issue is how a digital library service provider copes with an increase in its user base without deteriorating the quality of service and the effectiveness of database maintenance.

In the classical ISIS model, a database consists of a set of files represented by a master file that holds core data, and some auxiliary indexing files. Each set constitutes a centralized and independent database. Digital library providers that use ISIS-based repositories offer web access typically by binding a web server infra-structure to ISIS console tools through a CGI interface. That strategy is feasible and adopted to serve large databases in practice. The adoption of CGI interfaces is well-know, easy to implement and serves the purpose of serializing access to a non-multi-threaded ISIS back-end. However, the obvious shortcomings are sub-optimal performance, poor scalability and the existence of a single-point of failure.

One attempt to circumvent these issues is to deploy servers in parallel, serving a replicated database from an array of load-balanced servers. Although this solution is in production use, it is expensive, hard to maintain, and onerous to keep consistency across replicas.

Ideally, an efficient and scalable architecture should provide:

- low or no user-perceivable latency between some data input or update and the availability of this same data to end users;
- transparent access for end users;
- flexible maintenance;
- low data corruption risk or lost of consistency between replicas;
- decentralised topology to increase availability and avoid single point of failures;

The core of ISIS-NBP proposal is to tackle these issues, designing a flexible and network-aware architecture. Large and medium scale digital libraries providers will be the ones mostly benefited from this new design.

3.4 Increasing Interoperability

Concerning interoperability, the proposal encompasses a wide range of external interfaces, trying to maximize connectivity and interoperability. In the ISIS-NBP proposal we address the interoperability issue adopting an extensible set of standardised protocols. The rationale for this decision is given in table 1.

access method	advantages	disadvantages
local repository with proprietary API	- high data throughput - easy to implement - vertical scalability (improve server hardware)	- bottleneck to parallel simultaneous access - technology dependant - poor horizontal scalability - tight coupling between repository and applications
remote (or local) repository via proprietary protocol	- compatible to legacy - easy to implement	- little code reuse - start to implement from scratch - restricted interoperability
remote repository (or local) via open/standardised protocol	- maximum interoperability and code reuse - start from open implementations - loose coupling between repository and applications	- mapping protocol to repository model - evolution of standards (preserve standard compliance)

Table 1 – Repository Access Interfaces (Rationale)

The repository location transparency gives to digital libraries providers a lot of flexibility when it comes to deployment and management. The adoption of a standardised protocol brings the extra benefit of cross-library open-ended philosophy. Later in this report, we will present the set of protocols chosen to be initially supported by ISIS-NBP.

3.5 Migration Path

The ISIS-NBP project proposal aims to provide a flexible model, expanded capacity, better scalability and easier maintenance, plus a set of cross-platform and user friendly tools. Nevertheless, the classic ISIS and derived tools have a huge, an already trained user-base, and many systems in production use. Therefore, the key strategy to this endeavour success is adopting an evolutive and incremental approach, avoiding disruption of production systems and user-base.

To achieve a smooth transition path, ISIS-NBP will strive to maintain the new changes transparent for the current user base as long as possible. Some general guidelines for the migration path are:

- deploy ISIS-NBP in a controlled environment for validation, choose a small independent data provider close and accessible to the development ISIS-NBP core team.
- either support native access to classic CDS-ISIS databases or provide a transparent fully-automatic migration tool.
- support the ISIS formatting language and soften the learning curve to new features and tools.
- create user-friendly tools with support to network update and bug-report capabilities.

Later in this report we will present a more detailed migration strategy.

4 ISIS-NBP Proposal

In this section we propose solutions to the motivations presented in section 3. The core of the ISIS-NBP proposal consists of:

- rethinking CDS/ISIS model and moving towards a XML-based data model
- adopting a mixed storage strategy, accommodating from traditional ISIS master files to full Native XML Databases
- improving scalability through the adoption and deployment of operational units called ISIS Cells
- increasing interoperability by supporting several access protocols
- providing a migration path to avoid a disruptive technological transition

4.1 Rethinking the Model

We have pointed out the following limitations regarding model: expressiveness, capacity, and interoperability with external XML-based data providers and consumers. There is single measure that solves all limitations at the same time -- adopting a XML-based hierarchical model for ISIS-NBP:

- XML was designed to facilitate the sharing of data [22] across different information systems. XML has become the *de facto* free-fee open standard for data interoperability. Therefore, adopting a XML based model eases the path to seamless interoperation to external data providers and consumers
- The XML model is extensible and more legible and expressive, because it allows arbitrary nesting and mnemonical markup;
- XML is data encoding aware, solving several problems with multi-language support and internationalization (i18n);
- Abandoning a rigid binary layout for the start/end tag-based delimiters of XML we escape from the capacity limitations that stem from the number of bits chosen a *priori*. XML data streams are limited only by the underlying storage capacity.
- There are off-the-shelf cross-language tools that support parsing documents and checking for well-formation and validity
- XML is ubiquitous and being adopted in many domains of human knowledge to structure and represent information

In spite of all these characteristics, XML is often criticized for its: complexity, efficiency and verbosity. As for complexity, XML is no more complex than the domain it is trying to capture and represent, at least under the supposition of a clean design. As for efficiency, the ubiquitous adoption of XML stimulated Academia and Industry to develop faster and more mature tools, such as parsers, validators and even native databases. Therefore, remains the undeniable fact that XML is verbose. However, is there a real threat of storage and bandwidth costs increase due to this verbosity?

To answer that question we present the results of a simple experiment. We have taken the sample CDS/ISIS database with 150 records and converted it to both text and xml formats. The pure text format serves as a benchmark, since it has no meta-information overhead.

File Format	Size in bytes	Percentual increase	Compressed size in bytes	Percentual increase
cds.txt	59233	-	14709	-
cds.mst	62976	6.32%	18101	23.06%
cds.iso	72789	22.88%	19644	33.55%
cds.xml	67173	**13.40%**	14852	**0.97%**

Table 2- Comparison of storage consumption related to file format

From Table 2, we can draw two conclusions:

1. In spite of XML verbosity, even a naïve mapping from CDS/ISIS master file to a XML flat file presents equivalent storage consumption.
2. Adopting standard compression techniques, XML representation presents minimum overhead.

We have made attempts to fit the XML model into the classic ISIS model. Although feasible in theory, in practice there is too much complexity added. On the other hand, the reverse approach, consisting in mapping the classic ISIS model to XML is simultaneously feasible and simple.

Just to illustrate, we present a high-level, yet naïve, mapping from CDS/ISIS to XML:

- each record becomes a XML tree with root: <record mfn="*1*" status="Active">...</record>
- each entry in the directory becomes a child node: <field tag="*42*">some data</field>
- Repeatable fields could be materialised just by repeating child nodes with the same value for the attribute *type*.
- Sub-fields can be nested: <field tag="30"><subfield tag="a">1970</subfield> ... </field>

There are other possible mappings from CDS/ISIS records to XML constructs. The exact mapping is still under discussion and evaluation, thus it will be specified *a posteriori*. At this time we have identified the following desirable properties:

- the ISIS-NBP will handle *digital objects* [14], in opposition to records in CDS/ISIS
- each digital object will hold arbitrary and self-descriptive core data and metadata [17]
- digital object's content can be serialized to a XML representation
- each digital object has a unique ID, that identifies the object's collection, data provider and location across the network
- digital objects can be organized in a hierarchical fashion inside the repository, as well as the digital object's content internal organization can also be hierarchical
- a digital object or its content can be identified by a path expression that represents the position of the object or internal attribute in its contextual hierarchy

ISIS-NBP model and architecture should provide an adequate infra-structure to build the following aggregated services:

- policy-based access control
- data object locking
- transactional access to a collection of data objects
- versioning of data objects
- monitoring of data object access pattern, optional triggering pre-defined actions
- browsing, searching and text-based information retrieval (IR) tools

The incorporation of a information retrieval tool set to ISIS-NBP is a big leap from the boolean search model supported by CDS/ISIS, moving towards the services provided by modern global search engines services such as Google and Live Search. We enumerate a few candidate improvements to be incorporated to ISIS-NBP search capabilities and indexing sub-system:

- case insensitive e accent insensitive search
- indexing by: thesauri (synonym and antonym) and soundex (phonetic resemblance)
- use of stemming and lemmatization techniques
- *document clustering* to support exploratory search and recommendation
- results ranking

4.2 Rethinking the Storage

The academic literature [11] considers the following alternatives to build efficient XML data repositories:

Approach	**Description**	**Characteristics**
Flat streams	store the xml data stream as it is in the file system or in a BLOB (binary large object) in a database management system.	- fast for whole document retreival - slow for subset retrieval (needs parsing) - no default transaction when stored directly in the file system
Meta-modeling	Decompose XML structure and define a mapping to relational tables and columns [4, 5, 6]	- fast attribute accessible - full transactional supporting - slow to reconstruct the document in textual form (due to many join operations)
Mixed mode (redundant)	Use flat stream and relational mapping simultaneously, replicating the data. Data access uses the fastest mechanism depending if it requires whole or part.	- fast retrieval - slow update - overhead to keep consistency - consumes double of the storage

Approach	Description	Characteristics
Mixed mode (hybrid)	Use a document-size threshold to decide when to migrate from flat stream to relational mapping	- complex implementation and management - requires domain dependent calibration
Binary XML	Use a binary XML serialization instead of a textual representation. Several proposals: VTD-XML, BiM, Fast Infoset, CBXML, BXML-CWXML, WBXML, etc	- more compact representation - lacks *de facto* standard - complex implementation and management

Table 3- XML repository strategies

The ubiquitous adoption of XML technologies is pushing the Academia from these approaches into the investigation of native *repositories* for XML. There are many research proposals [7, 8, 9, 10] and a few commercial systems. Native XML Repositories are a must when:

- XML streams are very large in size but only small parts are needed and retrieved often
- One needs transactional support *and* fast retrieval to a subset of attributes and the format is unpredictable enough or deep enough that a relational mapping is not feasible.

If none of the above conditions are met, in addition to the discussion above, there are other advantages to adopt the **file system** as the XML repository:

- full-text searching is faster
- XML stream retrieval is usually a lot faster and more flexible than anything else.
- there are partitioning schemes that favour large files or small files, choosing partition type can be used to optimize performance
- off-the-shelf and ubiquitous compression techniques can be used to deal with storage consumption
- source code versioning control systems can be adapted to do versioning on xml streams
- several file-based management tools can be used or adapted to browse and manage the repository
- policy-based file-system access control can be reused to prevent malicious or accidental data corruption
- locking mechanisms can be used to allow access in parallel safely
- simple model and cross-platform implementation
- hard links and soft links can be used to efficiently provide different organizational data views
- overall easy and cheap maintenance

According to our incremental migration strategy, we recommend to start with a flat stream model. The adoption of a native XML repository demands careful evaluation of the efficiency and robustness of the implementations, and an investigation of our user access patterns for tuning purposes. Nevertheless, the transition from a flat stream implementation to a native repository can be done transparently if we adopt a standard interface to access XML data from the beginning, transparently isolating the remaining components of ISIS-NBP from the storage layer.

Prior to reach a final storage design, we ought to schedule a thorough introspection analysis to answer the following questions:

- what was the growth rate of ISIS databases along the last few years ? Is there a pattern that allows future growth projections?
- which capacity-related attribute is the most critical bottleneck considering real user demand ?
- what are the maintenance operations that are critical to improve latency response to end users ?
- considering autonomous users of digital libraries tools in developing and under-developed countries, what is the average software and hardware platform adopted ? What are the expected upgrades for the near future?

This is an expensive and non-trivial statistical endeavour. Nevertheless, it is crucial to the success and adoption of the ISIS-NBP products and by-products. This endeavour has already begun, and BIREME's team is pioneering it.

4.3 Improving Scalability

Several systems and tools have been built upon the classic ISIS model since UNESCO release of Micro CDS/ISIS in 1985, not to mention the mainframe ISIS version released in the 60's. Among the several incarnations of ISIS tools we can mention:

- CDS/ISIS: a generalised Information Storage and Retrieval system.
- WINISIS: a Windows version that may run on a single computer or in a local area network.
- JavaISIS: a client/server component that allow remote database management over the Internet and are available for Windows, Linux and Macintosh.

- GenISIS: a tool that allows users to produce HTML forms for CDS/ISIS database searching.
- ISIS_DLL: a dynamic-link library that provides an API for developing CDS/ISIS based applications

We could probably cluster most of the available ISIS tools in 3 categories: console based tools, graphical user interface (GUI) based tools, and web based tools. These 3 categories are depicted in illustration 1. It is important to notice that tools from all 3 categories are still in production use today. Small-size users with legacy operating systems and low-end hardware still make use of the lightweight console-based tools, intermediate-size uses make use of GUI-based tools and huge-size users and data providers make use of a Web-based infrastructure to provide services based on ISIS technology. All of them are valid users and should be contemplated by the ISIS-NBP proposal.

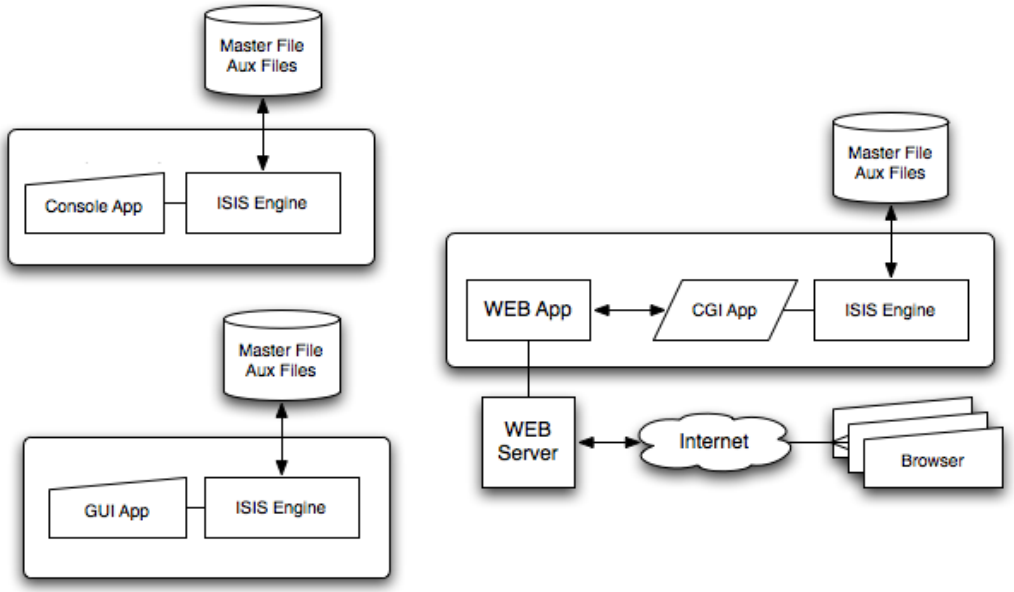

Illustration 1: Architecture of current CDS/ISIS tools

The design challenges faced by ISIS-NBP are:

- attend to the needs of small, intermediate and data providers;
- keep as much as possible a shared, portable (cross-platform), multi-platform (operating system independence) code base to decrease maintenance costs and increase robustness;
- minimize hardware demand, not forcing users to update their hardware prior to migrate to ISIS-NBP tools
- have equal or better performance than classic ISIS tools

We propose the concept of the ISIS-NBP Cell as the basic building block to compose scalable deploy scenarios and solve the aforementioned demands. As show in illustration 2, the ISIS-NBP Cell is an **atomic set** of tools that should be deployed all together to provide the minimum set of functionality.

The ISIS-NBP cell consists of:

- an embedded and self-managed **data repository**
- an **engine** providing a unified repository access API and a standard set of services
- a **console application** providing an user interactive environment
- a **server gateway** for remote access, cross-cell and external systems interoperability

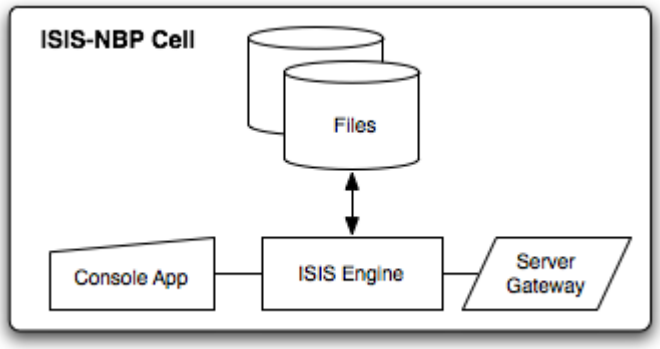

Illustration 2: The proposed ISIS-NBP Cell

The embedded repository will materialize improvements discussed in sections 4.1 and 4.2, shielding end users from managing themselves an external data repository. Moreover, by embedding the repository we create technical optimization opportunities once the Cell will have full-control over the repository.

The ISIS engine is critical to isolate the console and the gateway from the actual repository interface; therefore we can evolve repository technology incrementally and transparently. That will be critical to a successful migration. Furthermore, the engine encapsulates the services provided by the repository and interconnects the remaining modules of the Cell. The console application is vital for local maintenance of the repository, and should be present in every Cell deployed. The server gateway is a secure mechanism to make the Cell a jigsaw piece of distributed and inter-networked compositions, through a set of standardised supported

protocols. Therefore, it makes any ISIS-Cell an open-ended interoperable unit. We will discuss the supported protocols in section 4.4.

Now, we will build several deploy scenarios starting from the ISIS-NBP Cell, thus demonstrating how does it solve the scalability challenges.

4.3.1 Scenario 1 – Coupling to Applications

The basic scenario is the coupling of an ISIS-NBP Cell to applications. This is depicted in illustration 3. The same set of application categories can be coupled to the Cell: console based, GUI-based and Web-based. Some of the advantages gained by using the Cell are:

1. Applications are lighter because they only have to add value to user services and interface with the Cell through one of the supported protocols provided by the gateway. Therefore, they are faster to implement, more robust and easier to maintain.
2. Applications can be built to access a local Cell (same machine), but they can be **transparently** (without any modification to application code) routed to access a remote Cell (local network or across the Web) provided there is network connectivity.
3. More than one application instance (from any category) can access the same Cell in parallel. For instance, while an operator is accessing some Cell locally through a GUI-based application, the same Cell can be simultaneously accessed through a remote console application from which a power user is upgrading inner components or performing maintenance. This is possible because the server gateway component is responsible to serialise parallel access.
4. Pre-existing applications can be adapted to transparently talk to an ISIS Cells instead of directly handling a ISIS data files. Therefore, traditional and new tools can co-exist in the same working environment without data duplication or any risk of inconsistency.

We advocate that new applications should focus in web-technology instead of the traditional desktop GUI model. Even though desktop interfaces usually provide faster user response, they are tightly coupled to the underlying operating system or GUI middleware. Operating System and GUI middleware independence is a desirable goal. Moreover, web-based applications are much more flexible and provide the following benefits:

- they are naturally cross-platform and operating system independent.
- they can be accessed remotely (through the internet or a local area network).
- they can be seamlessly deployed in off-line personal computers, provided that lightweight web-servers are installed as well.
- applications can be modelled either as thin-clients or fat-clients. The former concentrates the logic in the server making software upgrades easy and supporting low-end client machines. The latter concentrates the logic in the client machine allowing a scalable yet cheap server infra-structure.

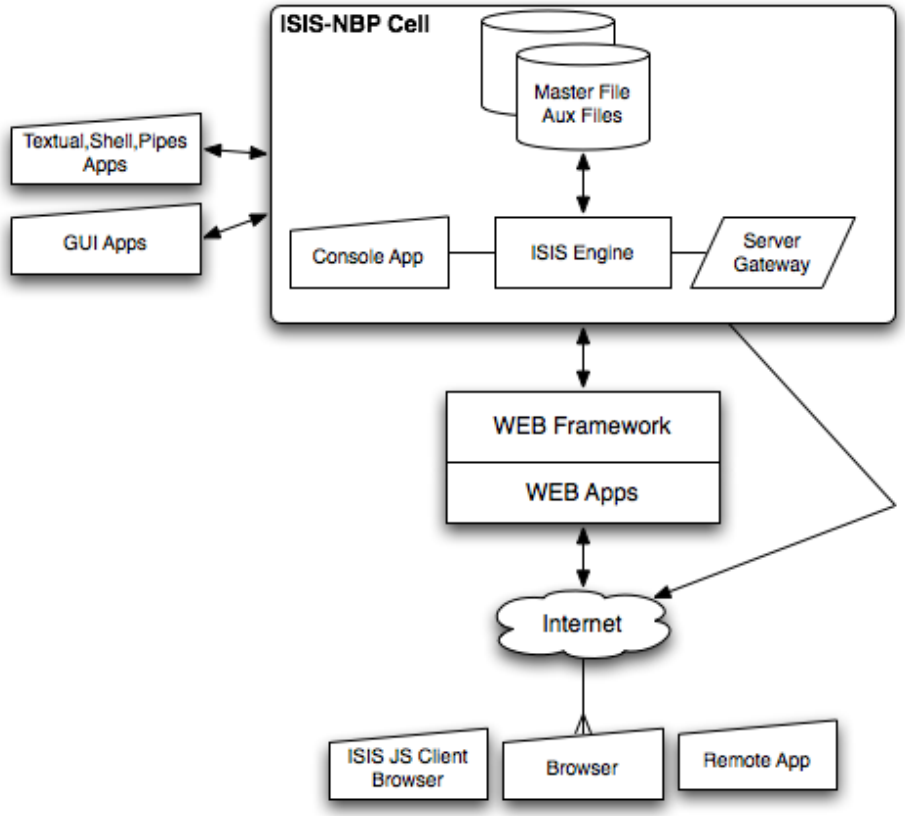

Illustration 3: Basic Scenario

4.3.2 Scenario 2 – Replication

Intermediate and huge net-based data providers, sooner or later, have to implement data replication, either to cope with the growth in user demand or to guarantee data preservation. Replication brings the issue of keeping replicas consistent. The scenario depicted in illustration 4 demonstrates a Cell composition to achieve transparent and horizontally-scalable data replication.

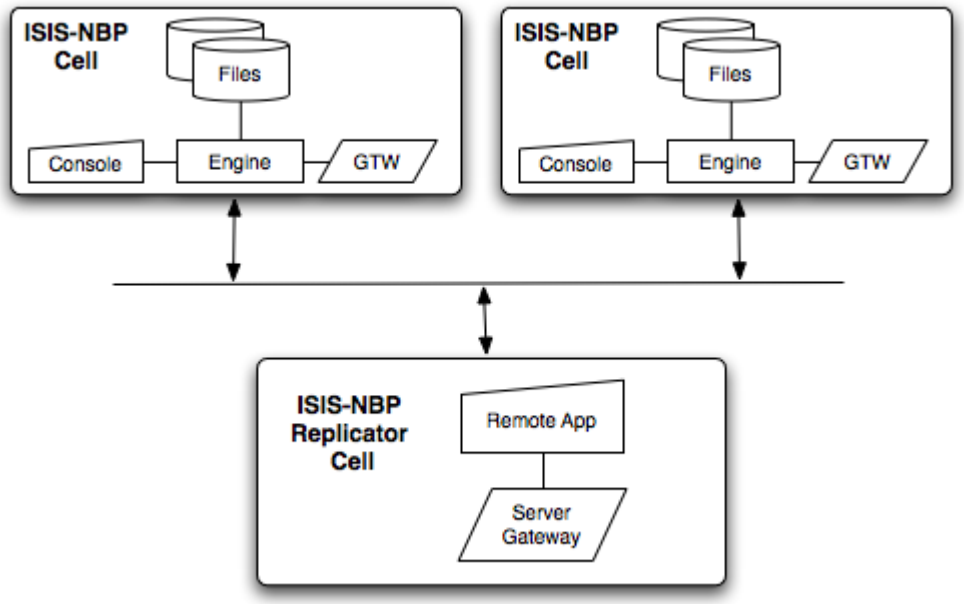

Illustration 4: Replication Scenario

In this scenario, all access is routed through the Replicator Cell that is responsible to keep the Cell Replicas consistent. It is important to notice that the Replicator Cell has the same interface supported by a generic ISIS-NBP Cell. Therefore, whoever talks to the Replicator Cell can be safely unaware of the fact that it is a Replicator.

4.3.3 Scenario 3 – Data Synchronization

Data synchronization is the process of exchanging information between two or more ISIS-NBP Cells. There are several modes of operation:

1. Given a target Cell, *export* all or a subset of its data to other Cells.
2. Given a target Cell, *import* all or a subset of the data contained in other Cells.
3. Given two target Cells, *equalize* their data collections by doing the necessary data imports and exports between each participant Cell.
4. Compare the differences between two or more Cells.

Illustration 5 shows a Cell designed to achieve any of the aforementioned tasks, performing the operation remotely across the Internet. Nevertheless, the same operation could be performed inside an Intranet or even in the same server machine. A Synchronization Cell would be useful to: fix problems with

inconsistent replicas, share data between different data providers, diagnose differences between different versions of data collections, etc.

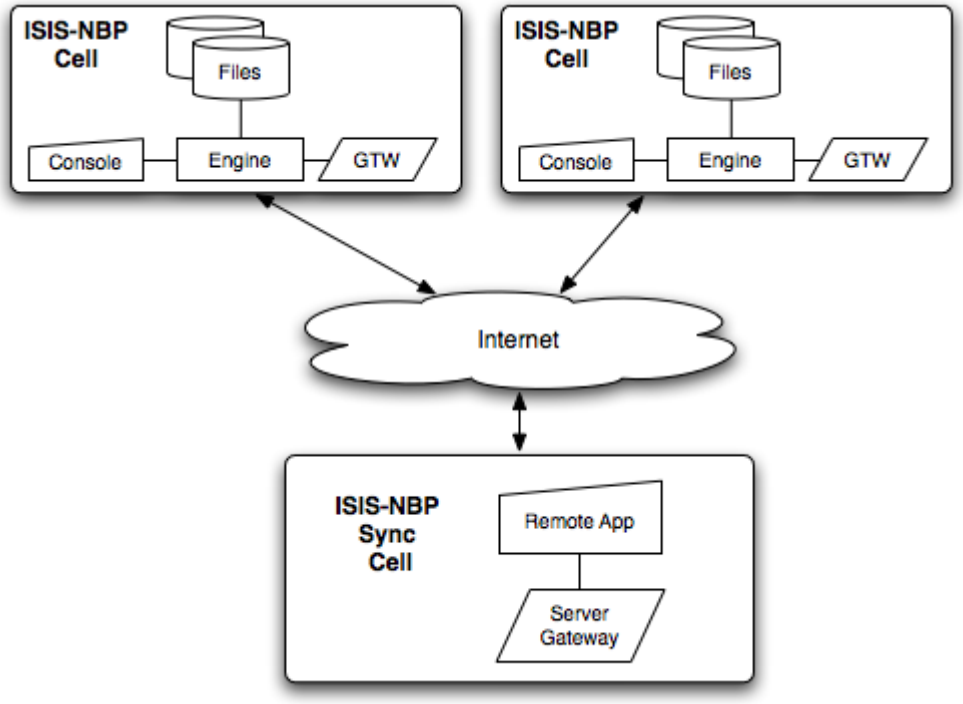

Illustration 5: Synchronization Scenario

One way to implement synchronization in an open ended-fashion is to adopt the SyncML protocol. SyncML (Synchronization Markup Language) is the former name (currently referred to as: Open Mobile Alliance *Data Synchronization and Device Management*) for a platform-independent information synchronization standard. However, SyncML might suffer some adpations prior to adoption, because today it is most commonly thought of as a method to synchronize contact and calendar information. In spite of data schema differences, SyncML structure and philosophy can be reused.

4.3.4 Scenario 4 – Smart Caching

One typical way to speed-up processing is caching data that is used often. The computer hardware and software already adopts caching strategies at several layers. Considering ISIS-NBP data providers that have a huge user base, a Smart

Caching Cell can be used to provide an extra level to speed-up data serving. A Smart Caching Cell could fast access to:

- raw records
- pre-formatted records
- query results

In the presence of a Smart Cache, the application server needs fewer resources to serve data requests, because it is much faster to simply fetch processed and formatted data from a cache than to: fetch raw data from the database, or reprocess and reformat raw data at every request, or even to re-run frequent queries.

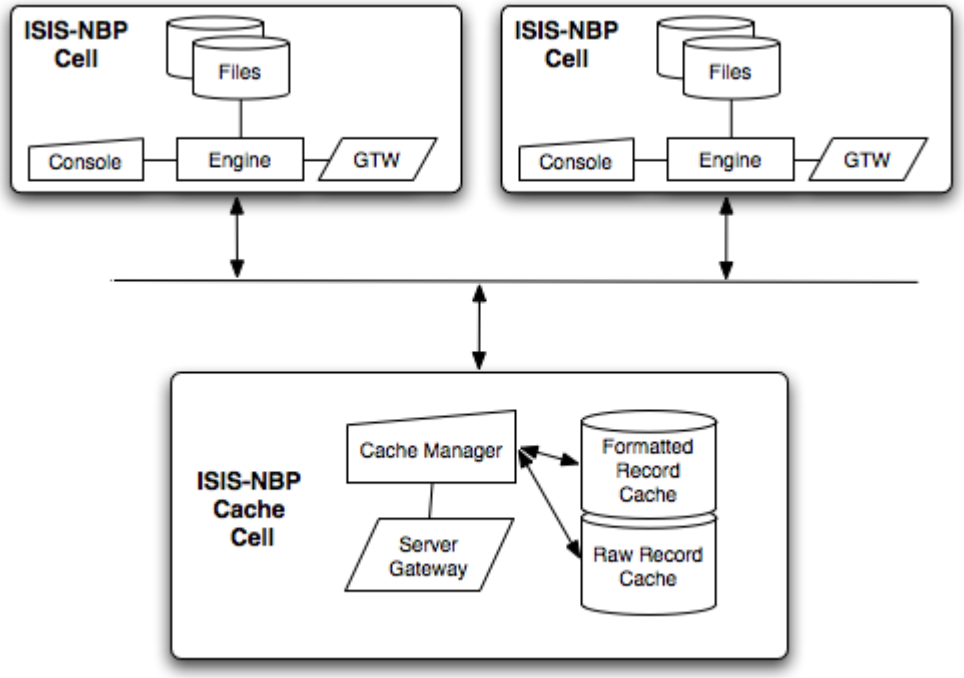

Illustration 6: Smart Caching Scenario

Caches implement as ISIS-NBP Cells are completely transparent to local or remote applications.

4.3.5 Scenario 5 – Data Versioning

Today there is a diverse plethora of version control systems. Considering the possibility of providing version support [25] for ISIS databases, that could be

achieved by interposing a Versioning Cell in between a generic ISIS Cell and any client application. If the application has no need for version support, it can simply access the Versioning Cell in the same fashion it would access any ISIS Cell. If the client application is version-aware, it can make a request for a specific version.

On each access request, the Versioning Cell can decide what policy to implement: delegate the request, return the most up-to-date version for the request or even return some specific version of data given the context of who made the request. Illustration 7 is an example of a Cell composition that supports versioning.

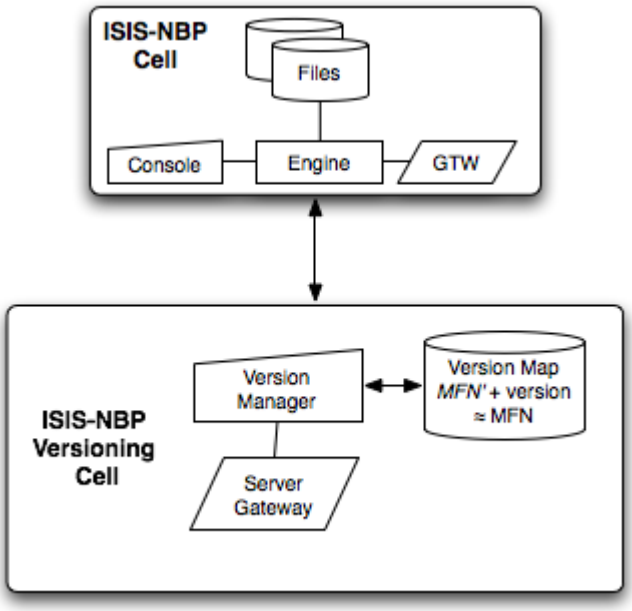

Illustration 7: Versioning Scenario

4.4 Increasing Interoperability

Interoperability is one of the key benefits from ISIS-NBP proposal. The server gateway module, inside de ISIS-NBP Cell, is the Cell's default interface to other Cells or applications. Therefore, the set of protocols supported by the server gateway module are a direct measure of ISIS-NBP interoperability [12].

No single protocol could achieve by itself the level of interoperation we desire to provide. So, we propose that ISIS-NBP Cells should support several protocols,

being able to handle them simultaneously. So far we have evaluated and proposed the adoption of the following protocols:

4.4.1 WebDAV: Web-based Distributed Authoring and Versioning

WebDAV is an IETF standard, similar to HTTP, that provides functionality to create, change and move documents on a remote server. Important features in WebDAV protocol include locking (overwrite prevention), properties (creation, removal, and querying of information about author, modified date, etc.), name space management (ability to copy and move Web pages within a server's namespace) and collections (creation, removal, and listing of resources).

4.4.2 RDF: Resource Description Framework

RDF [24] is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata model, used as a general method of modeling information through a variety of syntax formats. RDF's simple data model and ability to model disparate, abstract concepts has also led to its increasing use in knowledge management applications. Moreover, it is being considered the standard protocol to describe data in the Semantic Web.

4.4.3 RSS: Really Simple Syndication

Also known as, Rich Site Summary or RDF Site Summary, RSS is a family of web feed formats used to publish frequently updated digital content. RSS delivers its information as an XML file called an "RSS feed," "webfeed," "RSS stream," or "RSS channel".

4.4.4 OAI-PMH: Open Archives Initiative Protocol for Metadata Harvesting

OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) [19, 20] is a protocol developed by the Open Archives Initiative. It is used to harvest (or collect) the metadata descriptions of the records in an archive so that services can be built using metadata from many archives.

4.4.5 STOMP: Streaming Text Orientated Messaging Protocol

Streaming Text Orientated Message Protocol (STOMP), briefly known as TTMP, is a simple text protocol designed for working with Message Oriented Middleware. It supports transactional message exchange and a publish/subscribe pattern of peer communication. Due to the simplicity of its design it is very easy to develop clients and many exist for different languages and platforms.

This collection of protocols is our initial network-toolbox to empower the ISIS-NBP Cell with superior interoperability.

4.4.6 Web Service (Service Oriented Architecture)

There are also plans to support a *Web Services* [16, 20] complaint interface, including WSDL descriptions for XML-RPC and SOAP-based interoperability.

4.5 Migration Path

According to Michael Brodie and Michael Stonebraker in *Migrating Legacy Systems*, the steps to a smooth and successful migration are:

1. Analyze the legacy Information System (IS);
2. Decompose the IS structure;
3. Design target interfaces;

4. Design target applications;
5. Design target database;
6. Install the target environment;
7. Create and install gateways between the new and old environment;
8. Incrementally migrate legacy databases;
9. Incrementally migrate legacy applications;
10. Incrementally migrate legacy interfaces;

These 10 steps allow, as much as possible, to safely cut over to the target IS while phasing out the previous one.

In the first step, we assume that the specifics of day-to-day business operations might be out-of-date or non-existent. Therefore, producing a detailed, up-to-date documentation about the specifics of the legacy data and functionality is a labour intensive task. One possible course of action is to write tools to automatically extract specifications from the code base.

In the second step, decomposition granularity is defined: interface modules, procedure calls, database services. An important goal in this phase is ensure isolation, which means that modifications to program A should not have an effect on another program B.

The importance of the third step is to facilitate the transition into a distributed environment by maintaining a link to pre-existing applications and databases. Moreover, this step provides an early opportunity to verify the target IS ease of use, connectivity and portability. This is the step that will guarantee backward compatibility.

The forth and fifth steps are the phases where new functionality is materialised and prior limitations are overcome. An incremental approach here means to adopt short design-develop-test cycles trying to avoid the propagation of mistakes to the ulterior phases.

In the sixth phase the emphasis should be on testing and validation. This is when the value added by the new IS can be measured in terms of: user experience, maintenance costs and performance. Moreover, this phase serves the purpose to

document and troubleshoot the deploy process. This is the last opportunity to cheaply and undisruptively correct glitches or design flaws.

The seventh phase is what guarantees that legacy applications and databases in production use continue to function seamlessly in parallel to the target IS.

The remaining phases suggest an ideal order to phase out legacy data and tools. All participants agreed that this interface should be built over the existing file formats and include the features extensively used in the current WinISIS system and its format language.

5 Related Work

In the elaboration of the ISIS-NBP proposal, we have not only examined the needs of the ISIS Community, but we have done a brief investigation of relevant projects and efforts in the digital library domain. Several of these projects address issues that intersect with those addressed by ISIS-NBP; consequently we should pay close attention to the evolution of these projects and learn from their success as well as from their failures.

5.1.1 Fedora

Fedora (or Flexible Extensible Digital Object Repository Architecture) is a modular architecture built on the principle that interoperability and extensibility is best achieved by the integration of data, interfaces, and mechanisms as clearly defined modules. Fedora is a digital asset management (DAM) architecture, upon which many types of digital library systems might be built. Fedora is the underlying architecture for a digital repository, and is not a complete management, indexing, discovery, and delivery application. A Fedora Repository provides a general-purpose management layer for digital objects, and containers that aggregate mime-typed datastreams (e.g., digital images, XML files, metadata). Out-of-the-box Fedora includes the necessary software tools to ingest, manage, and provide

basic delivery of objects with few or no custom disseminators, or can be used as a backend to a more monolithic user interface. Fedora is developed jointly by Cornell University Information Science and the University of Virginia Library. The Fedora Project is currently supported by grants from the Andrew W. Mellon Foundation.

5.1.2 DSpace

DSpace is an open source software package which provides the tools for management of digital assets, and is commonly used as the basis for an institutional repository. It is also intended as a platform for Digital preservation activities. DSpace uses a relational database, and makes its holdings available primarily via a web interface, but it also supports the OAI-PMH v2.0, and is capable of exporting METS (Metadata Encoding and Transmission Standard) packages also.

5.1.3 Greenstone

Greenstone is a suite of software for building and distributing digital library collections. It is not a digital library, but provides a fully-searchable tool for building metadata-driven digital library.

Greenstone is interoperable using contemporary standards, supporting Open Archives Protocol for Metadata Harvesting (OAI-PMH), and import/export of metadata in METS and DSpace formats. Moreover, plug-ins are used to ingest externally-prepared metadata in different forms, such as: XML, MARC, CDS/ISIS, ProCite, BibTex, Refer. Plug-ins are also used to ingest documents, in several formats such as: PDF, PostScript, Word, RTF, HTML, Plain text, Latex, ZIP archives, Excel, PPT, Email (various formats), source code.

It has been developed and distributed in cooperation with UNESCO and the Human Info NGO in Belgium. It is open-source, multilingual software, issued under the terms of the GNU General Public License.

5.1.4 Milos

MILOS [13] Multimedia Content Management System is a general purpose software component tailored to support design and effective implementation of digital library applications. MILOS supports the storage and content based retrieval of any multimedia documents whose descriptions are provided by using arbitrary metadata models represented in XML.

MILOS has an advanced XML Search Engine, developed at ISTI-CNR that supports: XQuery, offers image similarity search, full-text search and categorisation. The MILOS project is supported by ECD and VICE, funded by MURST and DELOS NoE funded by the EU IST program.

6 Final Remarks

After presenting all of our arguments in favour of ISIS-NBP proposal, we are convinced that:

- the adoption of a XML-based storage will allow ISIS-NBP to overcome CDS/ISIS limitations in terms of capacity and expressiveness. Moreover, it will ease interoperability with other digital libraries and external software systems.
- ISIS-NBP Cells provide a flexible and scalable architecture to build digital library repositories and applications.
- the strategy of carefully planning a migration path, from the beginning, will ease the transition and co-existence in production use of classic CDS/ISIS databases and applications to become adherent to the model and tools offered by ISIS-NBP.

7 Bibliographic references

1. Unesco. Mini-micro CDS/ISIS Reference Manual versão 2.3. Unesco. Março 1989

2. J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. Technical report, Stanford University Database Group, February 1997. http://citeseer.ist.psu.edu/mchugh97lore.html

3. J. McHugh, J. Widom, S. Abiteboul, Q. Luo, and A. Rajamaran. Indexing semistructured data. Technical report, Stanford University, Computer Science Department, 1998. http://citeseer.ist.psu.edu/mchugh98indexing.html

4. Daniela Florescu and Donald Kossman. A performance evaluation of alternative mapping schemes for storing XML data in a relational database. Technical Report 3684, INRIA, March 1999. http://citeseer.ist.psu.edu/florescu99performance.html

5. P. Bohannon, J. Freire, P. Roy, and J. Simeon. "From XML Schema to Relations: A Cost-Based Approach to XML Storage". In IEEE ICDE, San http://citeseer.ist.psu.edu/465172.html

6. M. Ramanath, J. Freire, J. Haritsa, and P. Roy. Searching for efficient XML to relational mappings. Technical Report TR-2003-01, DSL/SERC, Indian Institute of Science, 2003. http://citeseer.ist.psu.edu/article/ramanath03searching.html

7. T. Fiebig, S. Helmer, C.-C. Kanne, J. Mildenberger, G. Moerkotte, R. Schiele, and T. Westmann. Anatomy of a native xml base management system. Technical Report 01, University of Mannheim, 2002. http://citeseer.ist.psu.edu/fiebig02anatomy.html

8. H. V. Jagadish, Shurug Al-Khalifa, Laks Lakshmanan, Andrew Nierman, Stylianos Paparizos, Jignesh Patel, Divesh Srivastava, and Yuqing Wu. Timber: A native XML database. Technical report, University of Michigan, April 2002. http://www.eecs.umich.edu/db/timber/ . http://citeseer.ist.psu.edu/jagadish02timber.html

9. A. Deutsch, M. Fernandez, and D. Suciu. Storing semi-structured data with stored, 1998. Manuscript available from http://www.research.att.com/~suciu. http://citeseer.ist.psu.edu/deutsch99storing.html

10. C.C Kanne, Guido Moerkotte. Efficient storage of xml data. Proc. of ICDE, California, USA, page 198, 2000. http://citeseer.ist.psu.edu/kanne99efficient.html

11. B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In Proceedings of VLDB, 2001. http://citeseer.ist.psu.edu/cooper01fast.html

12. David Bainbridge, Katherine J. Don, George R. Buchanan, Ian H. Witten, Steve Jones, Matt Jones and Malcolm I. Barr, Dynamic Digital Library Construction and Configuration, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

13. Giuseppe Amato, Claudio Gennaro, Fausto Rabitti and Pasquale Savino. Milos: A Multimedia Content Management System for Digital Library Applications, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

14. George Pyrounakis, Kostas Saidis, Mara Nikolaidou and Irene Lourdi. Designing an Integrated Digital Library Framework to Support Multiple Heterogeneous Collections, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

15. Unni Ravindranathan, Rao Shen, Marcos André Gonçalves, Weiguo Fan, Edward A. Fox and James W. Flanagan. Prototyping Digital Libraries Handling Heterogeneous Data Sources – The ETANA-DL Case Study, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

16. Ann Apps. zetoc SOAP: A Web Services Interface for a Digital Library Resource, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

17. Qiuyue Wang, Wolf-Tilo Balke, Werner Kießling and Alfons Huhn. P-News: Deeply Personalized News Dissemination for MPEG-7 Based Digital Libraries, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

18. Kurt Maly, Michael Nelson, Mohammad Zubair, Ashraf Amrou, Sathish Kothamasa, Lan Wang and Rick Luce. Enhancing Kepler Usability and Performance, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

19. Fabio Simeoni. Servicing the Federation: The Case for Metadata Harvesting, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

20. Sergio Congia, Michael Gaylord, Bhavik Merchant and Hussein Suleman. Applying SOAP to OAI-PMH, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

21. Amit Kumar, Alejandro Bia, Martin Holmes, Susan Schreibman, Ray Siemens and John Walsh. < teiPublisher>: Bridging the Gap Between a Simple Set of Structured Documents and a Functional Digital Library, Research and Advanced Technology for Digital Libraries. Lecture Notes in Computer Science. Volume 3232/2004. September (2004).

22. Peter Buneman. What the Web Has Done for Scientific Data – and What It Hasn't, Advances in Web-Age Information Management. Lecture Notes in Computer Science. Volume 3739/2005. October (2005)

23. Xiaoguang Li, Jian Gong, Daling Wang and Ge Yu. An Effective and Efficient Approach for Keyword-Based XML, Advances in Web-Age Information Management. Lecture Notes in Computer Science. Volume 3739/2005. October (2005) Retrieval

24. Jing Lu, Yong Yu, Kewei Tu, Chenxi Lin and Lei Zhang. An Approach to RDF(S) Query, Manipulation and Inference on Databases, Advances in Web-Age Information Management. Lecture Notes in Computer Science. Volume 3739/2005. October (2005)

25. Yuan Wang, Gang Chen and Jin-xiang Dong. MPX: A Multiversion Concurrency Control Protocol for XML Documents, Advances in Web-Age Information Management. Lecture Notes in Computer Science. Volume 3739/2005. October (2005)

26. Roy Tennant. XML in Libraries (2002)

Appendix I - CMS Interoperation

In this section we discuss a sub-goal of the ISIS-NBP proposal: to allow an easy coupling between ISIS-NBP and CMS, accelerating the development of applications by benefiting from CMS tools.

Even though there are many CMS in the market, we have chosen to compare ISIS against Plone. Plone is not only a very successful CMS, but it is an international community effort that follows FOSS practices and guidelines. The following table compares classic CDS/ISIS resources and operations against the Plone software stack[6].

6 **Plone is built upon CMF, Zope and Python.**

Classic CDS-ISIS	**Plone**
Single Storage Option: master file and auxiliary indexing files.	Multiple Storage Options: - FileStorage: single file (similar to ISIS master file) - **default** - DirectoryStorage: one directory per record, one file per revision - BerkleyStorage: hierarchical dictionary-based storage
Master File: Indexed Sequential Access Model (ISAM) with file system based direct access index file (.xrf)	FileStorage: ISAM with primary memory based direct access index file.
Record is self-descriptive (directory header)	Record is typed (type descriptions stored separated from data)
Specific language for: data query, extraction and formatting	data extraction: familiar syntax to any object oriented language, records are objects and fields are object attributes. formatting: general purpose programming language, HTML template system and CSS. Standardised tools for the Web domain.
Does **not** have native UNDO support in the database.	Have **UNDO** support in the database.
Text-oriented user interface support.	Web-oriented user interface support.
Inverted list based indexing (Btrees + postings)	Inverted list based indexing (Btrees) Full-text indexing (TextIndexNG).
Records and fields identified by ID(number). Sub-fields identified by single letter.	Records identified by ID(reference) or generic key string. Fields and sub-fields identified by name (string).
Two nesting level inside record: field and sub-field	Arbitrary nesting levels: field containers

Table 4- CDS/ISIS and Zope/Plone mapping

Not only this comparison evince that a mapping between classic CDS/ISIS and a modern CMS is feasible, but it is also desirable considering that a modern CMS (such as Plone) already solved many of the limitations that ISIS inherited from its earlier versions.

Plone is already being successfully adopted by digital library service providers. One by-product of the ISIS-NBP project will be a seamless integration path between

classic CDS/ISIS model and Plone. That coupling can be achieved very fast due to the overlapping structure of both.

Appendix II - Development Plan

Activities for the development of the ISIS-NBP application are as follows.

- Define Data dictionaries
  - Map ISIS to XML
    - Define layout and XML file organization on disc
    - Development of the ISIS – XML converter
      - Development of the XML – ISIS converter
    - Development of the CRUD (create, retrieve, update and delete) data management (DM) module
      - Development of the basic indexing module (inverted files using UID based on SHA1)
      - Development of the console application
    - Define the extended ISIS retrieval and format language (to accommodate metadata and hierarchy levels)
      - Development of the interface between the data management module and the retrieval module
        - Development of the network gateway
          - Development of the network adaptor to support WebDAV (map to CRUD access, REST filosophy)
          - Development of the network adaptor to support OAI-MPH
          - Development of the network adaptor to support RSS
          - Development of the network adaptor to support Web Services
            - Development of the remote console application that connects to the network gateway
              - Development of the cell client for Zope/Plone
              - Development of the cell client for PHP
              - Development of the cell client for Java
                - Development of the cache cell
                - Development of the sync cell
                - Adaptation of applications to connect to the ISIS cell
                - Development of the WinISIS equivalent application

Research implementations of Native XML DB solutions
- Development of a Native XML DB module
  - Migrate data to Native XML DB implementation
    - Adaptation of the data management and indexing module to interact with the Native XML DB module
    - Benchmark Flat Files and Native XML DB
- Indexing module optimization
  - Specify RDF metadata representation