DAR: Institutional Repository Integration in Action

Youssef Mikhail[1], Noha Adly[1,2], and Magdy Nagi[1,2]

[1] Bibliotheca Alexandrina, El Shatby 21526,
Alexandria, Egypt
{youssef.mikhail,noha.adly,magdy.nagi}@bibalex.org
[2] Computer and Systems Engineering Department, Alexandria University,
Alexandria, Egypt

Abstract. The Digital Assets Repository (DAR) is a system developed at the Bibliotheca Alexandrina to manage the full lifecycle of a digital asset: its creation and ingestion, its metadata management, storage and archival in addition to the necessary mechanisms for publishing and dissemination. In its third release, the system architecture has been revamped into a modular design including components that are best of the breed, in addition to defining a flexible content model for digital objects based on current standards and a focus on integrating DAR with different sources and applications. The goal of this paper is to demonstrate the building blocks of DAR as an example of a modern repository, in addition to discussing the challenges that face an institution in consolidating its assets and DAR's answer to these challenges.

Keywords: Institutional Repository, Integration, Modular Architecture, Digital Assets Repository, DAR.

1 Introduction

Given the explosion of digital content currently available and its variety, *Institutional Repositories* have become a vital and crucial component for any organization to preserve and manage the lifecycle of digital objects. Institutional repositories come in different varieties, however most of the current solutions are monolithic without enough flexibility to adapt their components as needs arise or they become too cumbersome to implement due to their complexity. Current solutions leave a lot to be desired in integrating applications that publish digital assets from the repositories in addition to integrating the repositories with different input sources. Bibliotheca Alexandrina (BA) developed its Digital Assets Repository (DAR)[1] to address these particular needs among other challenges that face repository administrators. DAR is an eco-system of components that manages the full lifecycle of a digital asset: its creation and ingestion, its metadata management, storage and archival in addition to the necessary mechanisms for publishing and dissemination. The design of DAR incorporates a modular best of the breed approach in different aspects of a modern repository. An API is provided that keeps applications in synch with the repository. Applications get the latest version of their digital objects or their metadata automatically once they are changed or added inside the repository. Several interfaces can be built on top of this API to integrate DAR with other systems thus extending its features.

DAR plug-in architecture provides flexibility to integrate with different sources of metadata, such as an ILS, other repositories or databases. This allows for collaboration with other repositories easily by ingesting some of their collections and synchronizing their metadata through plug-ins. DAR provides a flexible model to represent different types of digital objects. It also uses well established standards like METS[2] and MODS[3] for metadata. Usually there is a need to digitize an item before the metadata is ready. DAR allows the ingestion of the object in an intermediate state. Once the metadata is ready, the item becomes qualified for dissemination. After ingestion, the metadata of the object is kept in synch with the metadata sources, and can be updated by human operators if no metadata source exist.

Institutions also face a daunting problem of having several applications as separate silos where each application hosts a copy of the objects. This causes redundancy, divergence and failure to manage the original objects in a global consistent manner. DAR addresses this by managing one instance of the object inside the repository. DAR uniquely identifies objects using a persistent handle. Objects can be grouped into sets, and a single object can be a member of different sets. This allows administrators to share the objects among several applications where each application has access to particular sets of objects. Different applications can maintain different derivatives of this same object independently. DAR heavily relies on RDF relations to define sets and relations between objects.

This paper is structured as follows: Section 2 presents some of the related work. Section 3 gives an overview of the system architecture and sections 4 through 8 describe in detail the main system components. A scenario for integration in included in Section 9. Section 10 concludes and presents directions for future work.

2 Related Work

Due to the increasing need to preserve and manage the large amount of digital assets currently owned by institutions, several repository solutions have emerged. EPrints [4] provides a solution to preserve scientific data, theses, reports, and multimedia, with optimized support for Google Scholar focusing on open access research. Greenstone [5] is an open source software solution for building and distributing digital library collections. DSpace [6] is commonly used by academic and research libraries as an open access university repository for managing faculty and student output and has the largest installation base. DSpace offers a full application and not just a framework with built in full text search based on Lucene [7]. It uses qualified Dublin Core as the default metadata schema. The previous alternatives aim at providing an out-of-the-box complete experience and ease of adoption. However, detailed configurations and customizations require more effort and time and extensibility is limited. They do not offer the level of metadata representation flexibility, scalability, or modularity required for large repositories with very specific requirements. Fedora [8] is a conceptual framework that uses a set of abstractions for expressing digital objects providing the basis for developing software systems for searching and administration through the provided web APIs. In Fedora, content is managed as data objects, composed of components ("*Datastreams*") that contain either the content or metadata about it. It relies on a Content Model Architecture [9] to represent objects. Fedora does not provide an out-of-the-box solution, but rather requires programming expertise to setup the repository. Fedora's flexibility however makes it sometimes too cumbersome to implement in addition to requiring a certain level of programming expertise to build on top. DuraSpace [10] announced that DSpace and Fedora will be merging into DSpace with Fedora inside in 2011-2012, to retain the out-of-the-box experience that is DSpace, while also enabling the extra features that Fedora provides, e.g. versioning, relationships between objects, flexible architecture [11].

Stemming from the fact that most of the current solutions are monolithic without enough flexibility to adapt their components as needs arise or they become too cumbersome to implement due to their complexity, the search for the best solution for adoption has been the focus of many entities. The Arrow project [12] identified and tested solutions to support best practice institutional digital repositories for universities in Australia. It found that no one system or workflow will suit every university. Arrow recommended Fedora as a repository platform layer managing the object access and metadata combined with VITAL [13] as a services layer on top providing workflow extensions and advanced searching capabilities. The Hydra Project [14], a multi-institutional collaboration effort, on the other hand aims at providing an open source framework featuring the Fedora Repository on the back end, with a front end comprising Ruby on Rails, Blacklight [15], Solr [16], and a suite of web services providing similar functions. eSciDoc [17] also uses Fedora in its infrastructure to manage the digital objects while providing a set of loosely coupled open source services in a service-oriented architecture on top that can be used in building applications. The Stanford Digital Repository SDR [18] adopts Fedora in its Digital Assets Registry and as a metadata management system. *Digital stacks* comprise a suite of Ruby on Rails-based applications, with Solr index metadata stores providing asset discovery and delivery. SDR relies on Tivoli Storage Manager from IBM as its storage management subsystem. The SPAR project [19] at the BNF considers storing the Metadata using RDF in the Virtuoso triple store [20] the least risky approach [21] while still accepting a SIP with a METS manifest. It relies on iRods[22] for the storage subsystem.

DAR combines the best of the breed technologies to provide a modular repository based on latest standards with tools to manage digitization and encourage metadata entry by normal users. DAR data model is capable of representing all types of digital material. It focuses on the integration with different sources of metadata and objects through its flexible plug-in architecture, in addition to providing an API to integrate with applications that publish the digital objects stored within the repositories.

3 System Architecture

As shown in figure 1, DAR is divided into four main components: The *Digital Assets Factory* (DAF) provides a unified means of ingestion into the system from multiple sources through its ingestions plug-ins. A *Digital Assets Metadata* (DAM) subsystem manages the metadata even in an incomplete state. *Digital Assets Publishing* (DAP) components allow applications to synchronize objects and their metadata stored in their databases/indexes with the repository. Finally, the *Digital Assets Keeper* (DAK) manages access to the object files, versions and caching.

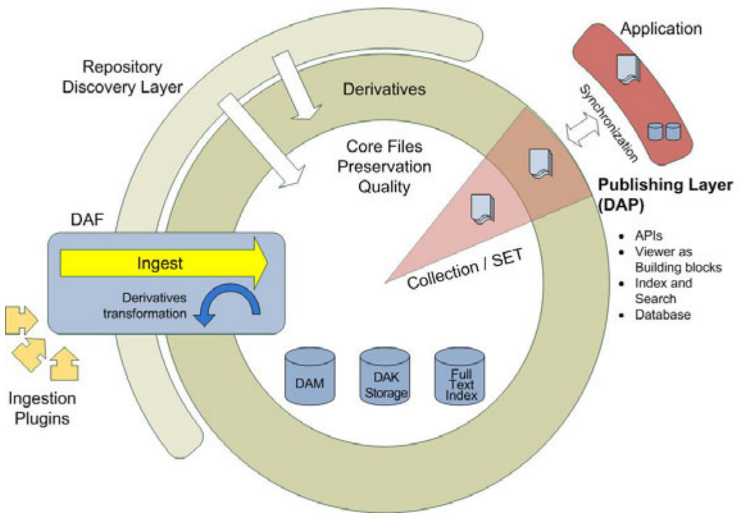

Fig. 1. DAR Conceptual overview

To facilitate object re-use, objects inside DAR are consolidated into sets or collections, and objects can belong to different sets. The core files, in preservation quality, are kept online on spinning drives for access by publishing applications. DAR also stores simple derivatives for all objects used for display by the *Discovery Layer*. Through the discovery layer, users of the repository can browse and search all assets stored within using simple viewers. A full text index based on Solr[22] search engine provides morphological search across the metadata and textual content of the objects stored in the repository.

Figure 2 depicts the detailed architecture of DAR in its latest version. In designing DAR, BA decided to rely on a modular design that allows it to mix the best of the breed components. The system incorporates components like Fedora, Mulgara[23], the Handle[24] system and search engines like Solr to manage metadata in a synergetic way. This modularity is possible through abstractions on the component level. A *RESTfull* API wraps the repository exposing its features to authenticated users. The API is used for ingestion of objects, metadata synchronization, discovery of items from the discovery layer and for application integration.

DAR uses METS to define objects stored within the repository. MODS is used for descriptive metadata for most of the object types. DAR uses Fedora as a *Metadata Registry* to manage the metadata. More information on that is provided in Section 5. Defining relations between objects is also important. RDF relations between objects are stored in Mulgara *Triple Store* providing facilities to run queries about set membership and object relations. Each object inside the repository is uniquely identified using a UUID. The UUID is used to generate a persistent Handle for that object making it possible to refer to the object consistently. A list of external identifiers are also stored with the object to provide references to the source of the object or to store any other form of identifiers that is specific to this type of object, e.g. ISBN for books.

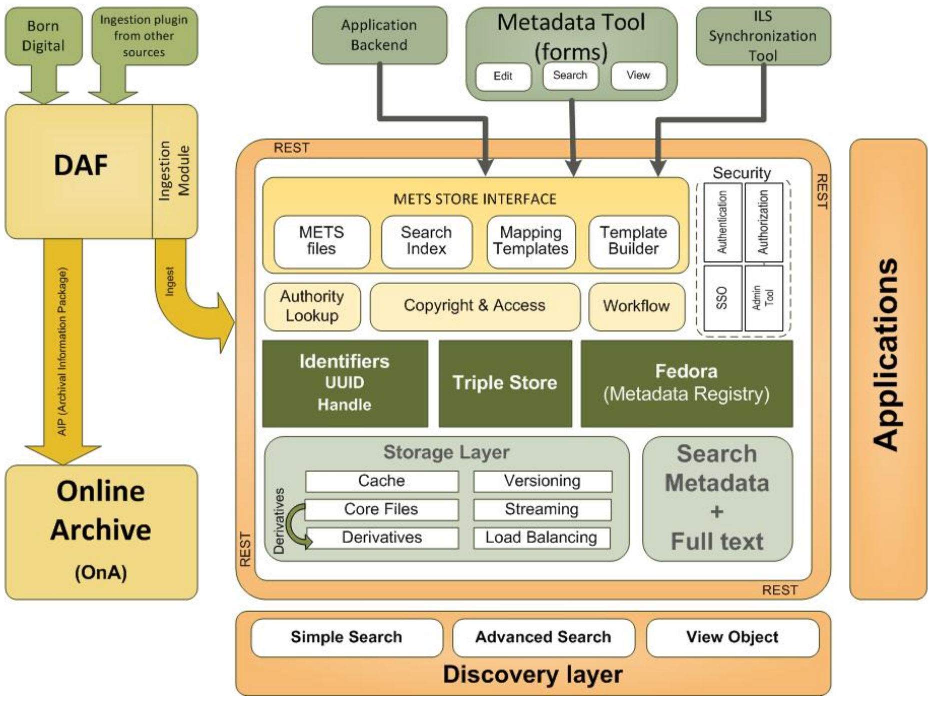

Fig. 2. DAR detailed architecture

A *Security* subsystem provides authentication and access control policy enforcement for specific sets or objects. DAR provides LDAP integration in addition to a local user database. A flexible authorization model enables administrators to define the different access levels for certain objects or sets. A *Single Sign On* module allows repository users to login once and gain access to all components of the system. A *Storage* subsystem provides a storage abstraction layer to isolate the underlying physical storage of the objects, in addition to other services like caching, load balancing among storage nodes, versioning and derivative management.

4 The Digital Assets Factory (DAF)

Many institutions are currently digitizing their collections to facilitate their access to users. Digital collections are very diverse and contain different types of objects. Since there is no one-size-fits-all tool that can perform all the digitization tasks, each object type might rely on a very different tool set and process to perform the digitization.

The Digital Assets Factory (DAF) [25] provides a configurable and flexible management tool for any digitization workflow, integrating with the current tools used for digitization. A digitization workflow is defined as a set of phases (figure 3), e.g. scanning, processing, quality assurance, encoding …etc. Administrators can define the sequence of phases required for a digitization in addition to adding pre-phase and post-phase checks, making sure that the process adheres to the digitization standards in the institution. DAF checks that the correct file types, number of files and naming conventions are in place before and after the current phase, and can manage several types of workflows for different object types. It integrates with automated phases offloading human operators to do tasks that humans are good at: e.g. OCR correction.

Understanding the importance of associating the object with its metadata as early as possible, DAF integrates with external sources of metadata, through the development of plug-ins, where the digital objects to be ingested have their metadata in an external ILS, repository or a database.

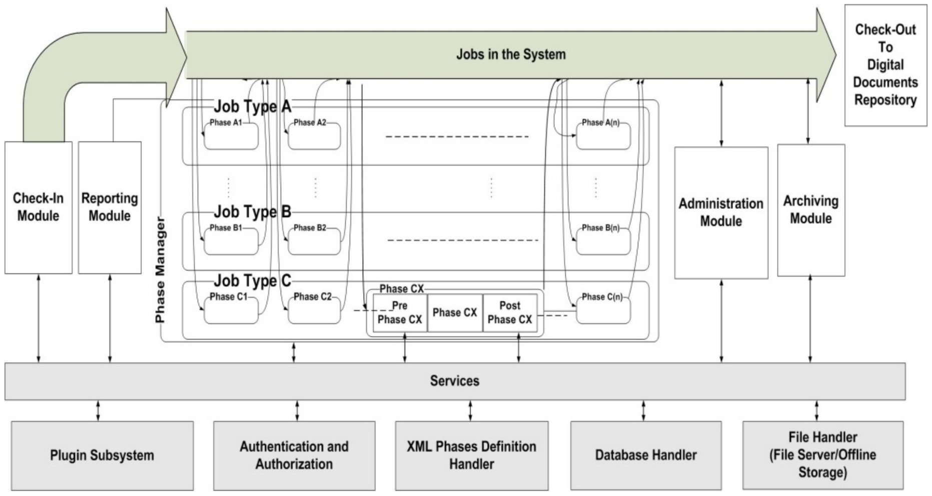

Fig. 3. DAF Architecture

Figure 3 demonstrates the system architecture of DAF: A *Check-in Module* adds jobs to the system. Operators will be able to work on these jobs through the various phases of the workflow managed by the *Phase Manager*. Each phase has a pre-phase and a post-phase check to verify the output. After the jobs complete the workflow, they are ingested into the repository through the *Check-out Module*, and an archival copy is sent automatically to the archive through the *Archiving Module*. If a job needs to be retrieved from the archive, DAF is used to fetch it and reload it into the system to perform the necessary actions upon it. DAF allows operators to request a re-do for a previous phase if they discover any problem. Operators can exchange messages indicating problems they face for a particular job so that other operators along the flow would take the necessary actions accordingly. The digitization supervisor checks the request, and can direct the job to another phase accordingly providing a means to correct the job as soon as the problem is detected. The quality of work and the performance of the digitization operators or automated phases can be monitored through the *Reporting Module*. DAF provides timely reports to various levels of management describing the workflow on a daily, weekly or longer basis allowing online queries about the current status of a certain asset during the digitization workflow. Reports display the number of jobs assigned to different phases of the workflow indicating the pending, running and finished tasks.

DAR is designed to be compliant with the OAIS [26] architecture since long term archiving of digital assets is crucial. DAF provides a unified means of ingestion into the system: in the pre-ingest stage, DAF handles both digital born objects and the digitization of physical objects. It creates the necessary SIP to be ingested into the repository, namely into the Digital Assets Keeper (DAK). DAF also creates an AIP that goes into the Online Archive (OnA) subsystem for long term archiving. Intermediate files created during the digitization are also included in the AIP for future reference. The item will be kept in sync with the metadata source and the AIP will be updated if any change occurs. BA provides DAF to the community as an open source tool (http://wiki.bibalex.org/DAFWiki).

5 The Digital Assets Metadata (DAM)

The wide spread and ease of use of technology has resulted in a plethora of different types of digital objects that an institutional repository should handle. Currently DAR holds more than 430,000 objects including books, photos, manuscripts, maps and documents with support for other types like videos, audio and more that are planned for ingestion. It is crucial to maintain the metadata along with the digital object. The object metadata are stored in Fedora, which is used as a metadata registry utilizing its different facilities to access the metadata.

5.1 The Content Model

DAR relies on well established standards to represent the metadata of the object, namely METS and MODS. However, it is a challenge to derive a model flexible enough to describe all types of library assets including books, maps, slides, posters, videos and sound recordings. DAR uses a hybrid *atomistic* and *compound* content model for objects providing flexibility of representation. An *atomistic* model represents every digitized piece of the object as a separate object, e.g. every photo of an album can be represented as a single object. Atomistic representation facilitates object re-use and discovery of a single piece of the object, e.g. a photo can be a member of several albums and can appear in a search result independently. Several objects can be aggregated into an *aggregate* object. An album containing multiple photos is an example of an aggregate object. The aggregate object holds metadata which is common across its child objects, e.g. album level metadata. When there is a need to create an album for photos taken in a certain conference grouped by conference sessions, photos of a certain session will be grouped into a session aggregate object, then session objects will be grouped into a conference aggregate object representing the conference event as a whole. There is no limit on the number of levels in this hierarchy.

The *compound* model represents the object as one single object containing the definition of the different digitized pieces of that object. Compound models greatly reduce the number of objects in the repository and thus the number of relations to be stored in the triple store making the maintenance of the object easier. DAR represents books as a compound object. Once the book is inserted into the metadata registry, it will contain a single data stream containing references to all pages, a convenient representation for a book since we will not be re-using its pages in another object. A single page would thus not be considered as a separate object. Figure 4 depicts the content model used for representing a book. A bibliographic record translates into a parent aggregate object to reduce metadata redundancy across multiple volumes. A volume (Book Y) is represented as an object bearing the volume level metadata. A compound object (Book Y Content) holds the actual content. It is worth noting that several books can be grouped into sets (Set X). A book can be a member of different sets. DAR provides the necessary tools to manage the set membership.

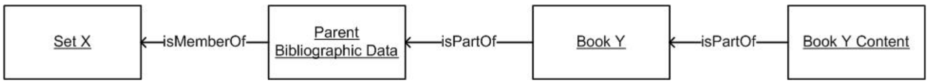

Fig. 4. A content model for books

The Fedora community defines a *compound* data model[9] as the representation used to define data objects containing different digitized pieces as multiple content-bearing data streams in a single object, e.g. a book containing multiple pages each represented as a separate data stream, or a book containing a PDF, DjVu, XML data streams. As described earlier DAR uses a slightly modified version of the compound model where there is a single content data stream containing the definition of the different pieces of the object. Gaining access to the pieces of the object is handled by DAR and not by the Fedora registry, thus making DAR independent from the Fedora-specific data stream representation of objects. This representation allows DAR more flexibility in handling the objects. DAR uses the same definitions for the *Atomistic* model as the fedora community.

5.2 The METS Store

The *METS Store* is a crucial component of DAM. When an object is ingested into DAR, a METS skeleton is first created storing the metadata provided upon ingest based on an XML template. The *METS Store* helps in handling one of the challenges faced by institutions, which is the need to quickly digitize some items for preservation where items are available for a limited time at the digitization facility with no or minimal metadata. The *METS Store* creates an intermediate incomplete state of metadata allowing the object to be ingested but the object is not yet ready for consumption.

Once the metadata is completed, either through synchronization with external metadata sources like an ILS or by a human operator, it is ingested into Fedora and the object is ready for usage through the publishing APIs. A simple yet flexible workflow engine handles these stages. Using this approach, a complete set of the metadata is kept in the *METS Store*, which acts as a backup for Fedora. DAR's modular architecture thus allows it to be independent of specific technologies. The metadata can be extracted from the *METS Store* and ingested into another system, and can also be used to reconstruct Fedora if any failure occurs.

5.3 Metadata Synchronization

The synchronization of the object metadata with external sources is based on XML templates to allow for flexible integration. An XML template, based on XSLT, is created to translate the output of the ILS or a database into the necessary MODS representation. This translation is handled by the *METS Store*. In case there are no sources of information to extract the metadata, human operators, assisted by authority lists, can complete the metadata through the use of dynamic forms in the *Metadata tool*. The tool generates human friendly metadata forms through configurable XML templates that suit the needs of the users. This tool is intended for use by normal users and not necessarily experienced librarians or catalogers. It displays a reduced version of the digital object to assist the user in filling the metadata. The XML templates define the necessary input fields, a friendly name for the fields in a particular project, input type validations for different data types and predefined list of values. The template also defines tool tips to assist the user in understanding what to be written in each field. The *Metadata tool* represents the objects in a tree hierarchy depicting the sets and objects within. When several objects share some metadata, the *Metadata tool* provides a facility to copy the metadata to objects in lower levels in the tree. It warns the users if conflicts or overwrites would occur.

The *Metadata tool* allows the operators to work on the object according to their tight schedules. An operator can chose to edit an object then save it for later completion. To ensure the quality of the metadata, the *Metadata tool* allows the designation of several roles: editors and reviewers. After the editor finishes his work, the object metadata is inspected and approved by a reviewer. Once approved, the metadata is considered ready for ingestion into Fedora. The workflow engine handles this review workflow. Users can browse their tasks whether pending editing, partially edited, pending review and tasks that have been approved. Full text morphological search based on Solr is provided to allow the operators to search across the metadata of the items in their collections or sets to retrieve an item for further editing.

5.4 The Copyright and Access Module

A flexible *Copyright and Access module* manages access to digital objects. Copyright information is stored with the object. An application provides the necessary authentication information to request access to its sets of objects in addition to the level of access required including viewing, downloading, printing, etc. The *Copyright and Access module* also coordinates the access of an object based on the number of licenses available. If the institution owns the right to display the object a certain number of times simultaneously, the module would coordinate the use of different applications to make sure the maximum number of licenses is not violated. This service is exposed through the REST API where applications attempt to obtain a license before displaying the objects. Based on the access policy associated with the object, a particular audience can be granted access to a partial view of the object, while a full view can be provided to another audience set.

6 The Digital Assets Keeper (DAK)

The Digital Assets Keeper is responsible for keeping a "working" copy of the object online used for consumption. A complete archival copy of the item is deployed into the *Online Archive* (OnA). DAK maintains a unique copy of the object and every object inside the repository is assigned a *persistent identifier*. DAK manages different versions of items. We are currently investigating the usage of pair-tree [27] structures among other approaches to provide a scalable and a flexible representation for object versions. A *storage abstraction layer* is used to isolate the repository from the underlying storage implementation. DAR requests access to an object through the object's identifier and a resolver would do the rest. This allows the implementation of different storage policies and several tiers of storage based on the frequency of use and other factors: e.g. frequently used objects can be kept on the fastest storage tier. The Storage layer handles load balancing across storage nodes in the same tier in addition to handling the caching of derivates used for the repository discovery layer and streaming of media files stored within the repository.

7 The Digital Assets Publishing (DAP)

There is a need to have applications highly integrated with the repository rather than being separate silos. Usually applications take a copy of the items, then they lose synch with the repository: they add, delete and modify the original objects without referring to the repository. DAR provides an API that allows applications to become repository-bound in a sense that when objects, that are in sets or collections the applications have subscribed in, are added, deleted or modified in the repository, the applications are notified and can request the latest updates of the object or the metadata through a RESTful API. One consistent instance of the object is kept in the repository and applications can focus on providing rich interfaces, creating their own derivatives of the objects while maintaining a link to the original object through the API and receiving updates.

Several applications have been built using this approach. A *Discovery layer* is provided for internal use inside BA giving the users access to the items in their totality inside the repository through simple viewers. Specialized *viewers* have been built to display items stored within the repository, such as books and photos. More viewers are still under development to provide unified access to the objects across applications built on top of the repository: e.g. tiled image viewer and manuscript viewer. *DAR books* (http://dar.bibalex.org) is an application built on top of DAR that displays the books stored in the repository (185,000 books) in a user friendly manner, providing browsing by facets, full text search and many other features. Whenever a book is added to or updated in DAR, it is automatically retrieved by *DAR books*. Another example is the print on demand (POD) integration layer that makes part of the content of DAR available through the POD system. Several interfaces can also be built on top of this API to integrate DAR with other systems.

8 The Online Archive (OnA)

The *Online Archive* (OnA) is a complete hardware and software solution that provides an underlying reliable and scalable archival storage. OnA ensures that any AIP ingested is mirrored at least once to provide redundancy. It heavily relies of checksums to ensure the integrity of the files at different stages. Given the exponential increase in the size of data to be archived, the OnA provides a low cost scalable storage system based on commodity hardware with spinning hard drives. It runs special software developed by BA for data management.

9 DAR Integration in Action

In the following section, we will introduce an example that describes the life cycle of a digital object in DAR. Suppose that a group of digital objects are part of a collection donated to the library by a certain organization X. The collection's metadata is availed through an OAI-PMH interface. The objects received still require further processing at the digitization facility at the library, e.g. image processing and OCR, then they are to be added to an already existing application Y that uses DAR's API.

A DAF plug-in is built to ingest the objects into the digitization workflow. Once the processing is done, the objects are archived and ingested into the repository in an intermediate state where a METS skeleton is built. Another Plug-in is developed to synchronize the metadata obtained through OAI-PHM with the *METS Store*. Once the synchronization is complete, the objects' metadata is ingested into Fedora, indexed and the archive is updated. The objects are now accessible through the API. Since these objects should be added as part of application Y, the administrator adjusts the set membership for these objects to application Y. The application therefore discovers the objects the next time it asks the API for updates and loads the objects to cache them on its servers. When the metadata of an object changes at organization X. The synchronization plug-in loads the new values. The *METS Store* and Fedora are updated and re-indexing of the object is triggered. Application Y detects that the metadata of the object is updated so it loads the updated values into its database and displays them.

10 Conclusions and Future Work

DAR is the flagship of Bibliotheca Alexandrina's digital library. We have presented in this paper DAR's architecture and the main philosophy behind its design. DAR addresses the main challenges faced by digital repositories including supporting different digital formats, digitization workflows, preservation of digital material and content dissemination.

At version 3.0, DAR's overall architecture and design are established. Most of its core components have been developed, using open source tools, and deployed with several applications launched on top. A complete data migration was completed in December 2010. Development is underway to enhance the Storage layer component and versioning, in addition to extending the *Copyright and Access module* and adding virtualization support. The potential of Linked Data, Semantic Web and triple stores is yet to be exploited further in DAR. Scalability issues related to the number of RDF relations are also been studied and other triple stores are currently being tested for scalability. BA is currently working on the migration of existing applications into the repository modifying them to be repository bound thus consolidating the digital assets.

References

1. Saleh, I., Adly, N., Nagi, M.H.: DAR: A Digital Assets Repository for Library Collections – An Extended Overview. In: Rauber, A., Christodoulakis, S., Tjoa, A.M. (eds.) ECDL 2005. LNCS, vol. 3652, pp. 116–127. Springer, Heidelberg (2005)
2. Metadata Encoding and Transmission Standard (METS), `http://www.loc.gov/standards/mets/`
3. Metadata Object Description Schema (MODS), `http://www.loc.gov/standards/mods/`
4. EPrints, `http://www.eprints.org/software/`
5. Greenstone, `http://www.greenstone.org/`
6. Dspace, `http://www.dspace.org/`
7. Apache Lucene, `http://lucene.apache.org/java/docs/index.html`
8. Fedora Commons, `http://fedora-commons.org/`
9. The Fedora Content Model Architecture (CMA), `http://fedora-commons.org/documentation/3.0b1/userdocs/digitalobjects/cmda.html`
10. DuraSpace, `http://www.duraspace.org/`
11. DSpace Fedora integration, `https://wiki.duraspace.org/display/DSPACE/Fedora+Integration`
12. The Arrow Project, `http://arrow.edu.au/`
13. VTLS VITAL software, `http://www.vtls.com/products/vital`
14. The Hydra Project, `https://wiki.duraspace.org/display/hydra/The+Hydra+Project`
15. Project Blacklight, `http://projectblacklight.org/`
16. Apache Solr, `http://lucene.apache.org/solr/`
17. eSciDoc, `https://www.escidoc.org/`
18. Cramer, T., Kott, K.: Designing and Implementing Second Generation Digital Preservation Services: A Scalable Model for the Stanford Digital Repository. D-Lib Magazine 16(9/10) (September 2010)
19. The SPAR Project, `http://www.bnf.fr/en/professionals/preservation_spar.html`
20. Virtuoso Universal Server software, `http://virtuoso.openlinksw.com/`
21. Fauduet, L., Peyrard, S.: A Data-First Preservation Strategy: Data Management In SPAR. In: iPres 2010 Vienna, Austria (September 2010)
22. iRODS, `http://www.irods.org/`
23. Mulgara Semantic Store, `http://www.mulgara.org/`
24. The Handle system, `http://www.handle.net/`
25. Yakout, M., Adly, N., Nagi, M.: Digitization Workflow Management System for Massive Digitization Projects. In: 2nd International Conference on Universal Digital Library ICUDL 2006, Alexandria, Egypt (November 2006)
26. Reference Model for an Open Archival Information System (OAIS), `http://public.ccsds.org/publications/archive/650x0b1.PDF`
27. Abrams, S., Kunze, J., Loy, D.: An emergent micro-services approach to digital curation in-frastructure. In: iPRES 2009, Mission Bay, San Francisco (October 2009)