

DAR: A Modern Institutional Repository with a Scalability Twist

Youssef Mikhail¹, 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 an Institutional Repository 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. DAR was designed with a focus on integrating DAR with different sources of digital objects and metadata in addition to integration with applications built on top of the repository. As a modern repository, the system architecture demonstrates a modular design relying on components that are best of the breed, a flexible content model for digital objects based on current standards and heavily relying on RDF triples to define relations. In this paper we will demonstrate the building blocks of DAR as an example of a modern repository, discussing how the system addresses the challenges that face an institution in consolidating its assets and a focus on solving scalability issues.

1 Introduction

Institutional Repositories has become an indispensable component in the arsenal of any institution that owns digital assets. The expansion of digitization efforts and the plethora of born digital materials have made it essential for an institution to maintain a repository that not only preserves the digital assets, but also manage their full life cycle and provide the users with the necessary tools to discover and use these assets. As the number of assets increase, several scalability issues arise that needs to be tackled including managing the ever expanding amount of storage required to store the objects, the efficiency of object representation and the need to rely on scalable RDF triple stores.

There are several options on the repository landscape to provide an institution with the necessary facilities to preserve and manage digital assets. However, most of the current solutions are monolithic without enough flexibility to adapt their components as needs arise. They also leave a lot to be desired in terms of integration with different input sources, in addition to application integration on top of the repository. Bibliotheca Alexandrina (BA) developed its Digital Assets Repository (DAR) (Saleh et al. 2005; Mikhail et al. 2011a; Mikhail et al. 2011b) to address these particular needs among other challenges that face repository administrators. DAR is an eco-system of components comprising a modular best of the breed approach 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.

DAR builds on well established standards like METS (Metadata Encoding and Transmission Standard, 2001) and MODS (Metadata Object Description Schema, 2005) for metadata. It provides a flexible model to represent different types of digital objects. DAR's 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. Further, usually institutions are faced with the need to digitize an item before its metadata is ready. DAR answers that by allowing the item to be ingested 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 exists. 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 (Handle system, 2011). Objects can be grouped into sets, and a single object can be a member of different sets. DAR heavily relies on RDF relations to define sets and relations between objects. This approach 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 while still maintaining a link to the source object in the repository. This is possible through an API that keeps applications in synch with the repository. Applications subscribe to particular sets. They use the API to 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 extending its features.

This paper is structured as follows: Section 2 presents some of the related work. Section 3 gives an overview of DAR architecture and the main system components. A scenario for application integration is included in Section 4. Section 5 describes a sample life cycle of a digital object. Scalability issues and how DAR addresses them are covered in Section 6. Section 7 concludes and presents directions for future work.

2 Related Work

There are several solutions on the repository landscape that provide facilities to preserve and manage the full lifecycle of the large amount of digital assets currently owned by institutions. Some of the repositories focus on providing facilities for certain types of digital assets and out-of-the-box experience like EPrints (EPrints, 2011) which provides a solution to preserve scientific data, theses, reports, and multimedia, with optimized support for Google Scholar focusing on open access research. Greenstone (Greenstone, 2011) is an open source software solution for building and distributing digital library collections. DSpace (Dspace, 2011) 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 (Apache Lucene, 2011). The previous solutions do not offer the level of metadata representation flexibility, scalability, or modularity required for large repositories with very specific requirements. Fedora (Fedora Commons, 2011) 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 (Fedora Content Model Architecture, 2007) 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 (DuraSpace, 2011) 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 (Diggory M., 2011).

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 (Arrow Project, 2008) 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 (VTLS VITAL software, 2011) as a services layer on top providing workflow extensions and advanced searching capabilities. The Hydra Project (Davis D., Green R., 2011), 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 (Project Blacklight, 2011), Solr (Apache Solr, 2011), and a suite of web services providing similar

functions. eSciDoc (eSciDoc, 2011) 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 (Cramer and Kott, 2010) 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 hosts several terabytes of assets and relies on Tivoli Storage Manager from IBM as its storage management subsystem that handles several tiers of data. The SPAR project (SPAR Project, 2011) at the BNF considers storing the Metadata using RDF in the Virtuoso triple store (Virtuoso Universal Server software, 2011) the least risky approach (Fauduet and Peyrard, 2010) while still accepting a SIP with a METS manifest. It relies on iRods (iRODS, 2011) for providing a scalable rule based storage subsystem.

The Fedora Performance and Scalability Wiki (Fedora Performance and Scalability Wiki, 2009) discusses known performance issues in Fedora Commons installations involving large-scale applications handling huge amounts of data like ingest performance and triple stores. The wiki gathers data, document limits and constraints for Fedora Commons installations. Interesting problems arise when there is a huge amount of items in a repository. HathiTrust describes an interesting finding in their full text index. The HathiTrust hosts over 5,000,000 digitized book titles in over 400 languages. They tested indexing the full text of 550,000 documents in 200 languages. Since the OCR varies in quality producing what is called as dirty text, where the text is not 100% correct, the number of distinct words recorded in the index exceeded 2.1 billion unique words in the index hitting the limit of their Solr search engine: a situation that had to be corrected by developing a Solr patch to remove this restriction (Burton-West T., 2010a). More problems appeared as they added more documents. The index becomes larger and slower (Burton-West T., 2010b).

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 efficiently. It integrates 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. DAR also provides solutions to some of the repository scalability problems.

3 DAR Architecture

Figure 1 shows the conceptual overview of DAR. The design is OAIS (Open Archival Information System reference model, 2002) compliant and is divided into four main components: The *Digital Assets Factory* (DAF) (Yakout et al, 2006) 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. The *Digital Assets Keeper* (DAK) manages access to the object files, versions and caching. Objects inside DAR are consolidated into sets or collections to facilitate object re-use, and objects can belong to different sets. DAR stores simple derivatives for all objects used for display through the *discovery layer* while caches the core files, in preservation quality, for access by the publishing applications. 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 search engine provides morphological search across the metadata and textual content of the objects stored in the repository.

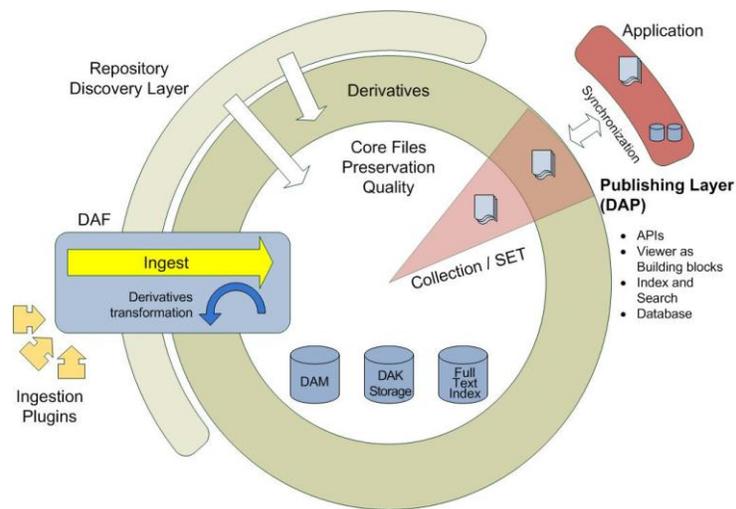


Figure 1. DAR Conceptual overview

Figure 2 depicts the architecture of DAR. The modular design of DAR allows for the incorporation of components like Fedora, 4Store (4Store triple store, 2011), the Handle system and Solr to manage metadata in a synergetic way. A RESTfull API wraps the repository exposing its features to authenticated users. The API is used for ingestion of objects, metadata synchronization, and 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. RDF relations between objects are stored in 4Store *Triple Store* providing facilities to run queries about set membership and object relations.

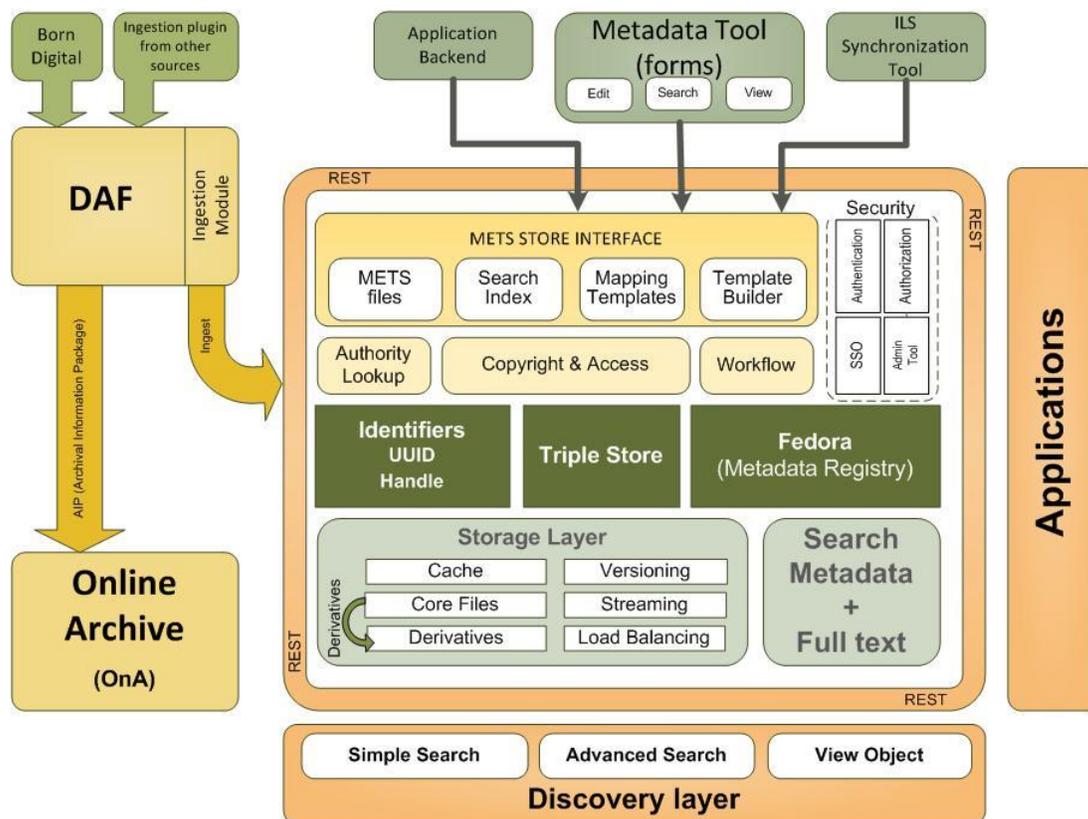


Figure 2. DAR detailed architecture

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. 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. 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.

3.1 The Digital Assets Factory (DAF)

Dealing with the digitization and ingestion of different types of objects is a daily challenge that faces repository administrators. DAF provides a configurable and flexible management tool for any digitization workflow where several workflows can be configured for different types of digital objects. When digital objects to be ingested have their metadata in an external ILS, repository or a database, DAF integrates with these external sources of metadata through the development of plug-ins.

Digitizing different types of objects requires the incorporation of several tools, and there is no one-size-fits-all tool that can perform all the different digitization tasks, where each object type might rely on a very different tool set and process workflow to perform the digitization. DAF integrates with these tools to assist repository administrators in managing the workflow. As shown in Figure 3, DAF divides each digitization workflow into phases. It can integrate with automated tools and scripts, checking their status at each phase and verifying their output through *pre-phase* and *post-phase* checks. DAF also makes sure the output is compliant to the digitization standards in terms of file types, number of files and naming conventions thus the human operators to do tasks that humans are good at: e.g. OCR correction. A *Reporting Module* provides timely reports about the status of the digitization operators and the automated tools.

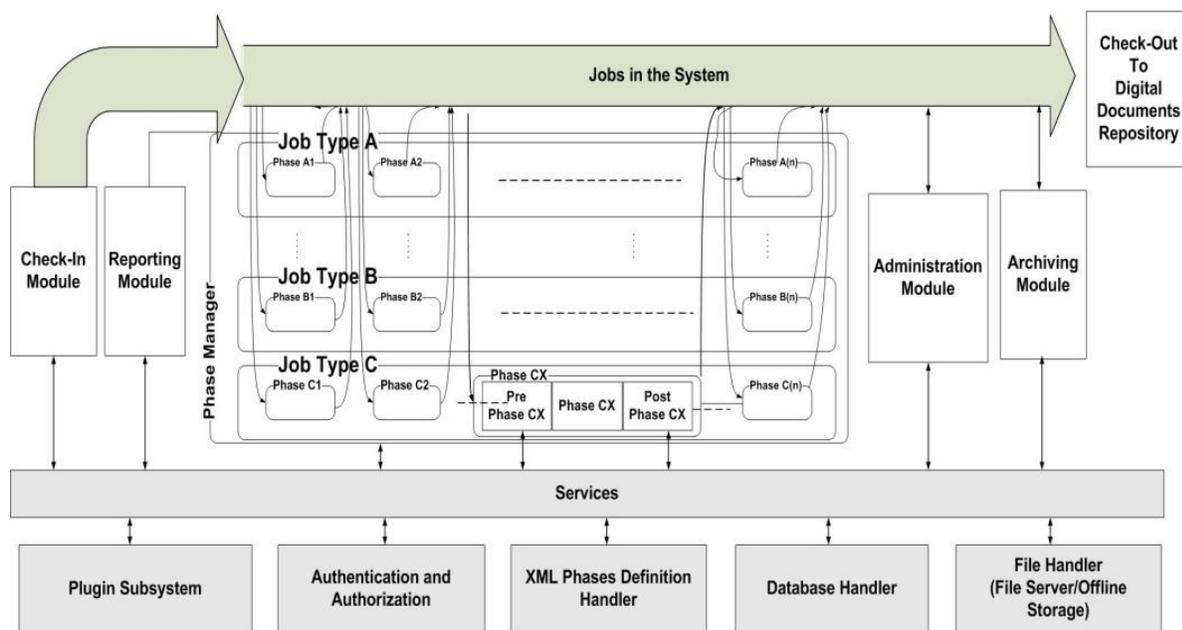


Figure 3. DAF architecture

DAR is designed to be compliant with the OAIS 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. The *Check-out Module* creates the necessary SIP to be ingested into the repository, namely into the Digital Assets Keeper (DAK). The *Archiving Module* 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. Once ingested, the item will be kept in sync with the metadata source through the Digital Assets Metadata subsystem.

BA provides DAF to the community as an open source tool (<http://wiki.bibalex.org/DAFWiki>).

3.2 The Digital Assets Metadata (DAM)

DAR relies on well established standards to record the metadata of the object, namely METS and MODS. The object metadata are stored in Fedora, which is used as a metadata registry utilizing the different facilities to access the metadata. Currently DAR holds more than 450,000 objects including books, photos, manuscripts, maps, 3D objects and documents with support for other types like videos, audio and more that are planned for ingestion. DAM stores RDF relations between objects in a *triple store*. Set membership is an example of the relations stored in the triple store. Fedora also uses the triple store to store its relations.

One of the challenges faced by institutions is the need to quickly digitize some items for preservation that are available for a limited time at the digitization facility with no enough metadata. The *METS store*, a crucial component of DAM, acts as an intermediate layer: when the object is ingested into DAR, a METS skeleton is first created. The metadata provided upon ingest is stored within this skeleton, which can be minimal. Once the metadata is complete through synchronization with other sources like an ILS, database or by a human operator, it gets 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 happens. 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, translates the output of the ILS or a database into the necessary MODS representation. This translation is handled by the *METS Store*. A template can be created for any source that requires integration.

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. The *Metadata Tool* provides several facilities to make it easy to use by regular users. Tooltips describe the usage of each field assisted by input validation and configurable authority lists. It represents the objects and their sets as a tree allowing users to copy metadata to objects down the tree sharing the same values. To ensure the quality of the metadata, the administrator can assign editors and reviewers for a particular set of items. Reviewers will provide quality assurance for the metadata before the record is considered complete and ingested into fedora. A workflow engine handles the different states of the metadata record approval, whether pending editing, partially edited, pending review and approved tasks. Users can save their work and return back to complete the record at their convenience. A full text morphological search is also provided to help editors and reviewers locate the records to be processed.

A flexible *Copyright and Access module* manages who can access the object and what level of access is provided, based on the application or user requesting access. Levels of access include: e.g. view, partial view, download, print, etc. Copyright information is stored with the object. If the object belongs to different sets and several applications can access this object and if institution is allowed to display the object a certain number of times simultaneously, the *Copyright and Access module* coordinates the access of the object by different applications denying requests for display if the limit is exceeded. Applications will request access through the REST API to obtain a license for display.

DAR uses a hybrid atomistic and composite content model of the 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 is represented as a separate object. Multiple photos are aggregated into an aggregate object. Album level information is stored in the aggregate object thus reducing the metadata redundancy. Whenever convenient, aggregate objects can themselves be aggregated with no limit on such hierarchy. The atomistic model facilitates objects re-use and discovery, where the object will appear in a in the results of a search query independently. The compound model represents the object as one single object containing the definition of the different digitized pieces of that object. This model greatly reduces the number of object stored in the repository and thus the number of relations between the objects. In DAR, a book is represented as a single compound object with one data stream containing references to all pages. Gaining access to these pages is handled by DAR and not by the Fedora registry. A page is thus not considered as a separate object. This is a convenient representation of a book since we will not reuse the page in another book. A book can be a member of different sets. Figure 4 depicts the content model used to represent 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).

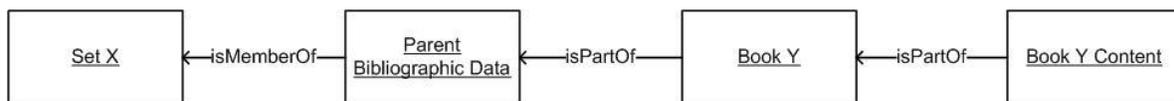


Figure 4. The content model for a book

3.3 The Digital Assets Keeper (DAK)

The Digital Assets Keeper (DAK) is responsible for keeping a “working” copy of the object online. It acts as a cache holding the objects components that are 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*. 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 derivatives used for the repository discovery layer and streaming of media files stored within the repository. DAK manages the versioning of items. We are currently investigating the usage of pair-tree (Abrams et al. 2009) structures among other approaches to provide a scalable and a flexible representation for object versions.

3.4 The Online Archive (OnA)

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 running special software developed by BA for data management. It 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.

3.5 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 enables the applications harness the benefits of integration with the repository. The applications 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 get a notification and can request the latest updates of the object or the metadata through a RESTful API. This allows keeping one consistent instance of every object in the repository. Maintaining a link to the original object in the repository through a well defined API would make it easy for applications to provide rich interfaces, getting updates for objects while still creating their own derivatives of the objects.

Several applications have been built using this approach. A *Discovery layer* is provided for internal use inside BA that gives the user access to the items in their totality inside the repository through simple viewers. Another example is the print on demand (POD) integration layer that makes part of the content of DAR available through the POD system. More interfaces can also be built on top of this API to integrate DAR with other systems.

Specialized *viewers* have been built to display items stored within the repository, such as books and photos. Several other 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.

4 A Case Study for application integration: DAR Books

DAR books (<http://dar.bibalex.org>) is an application built on top of DAR that displays the books stored in the repository. DAR Books serves 200,000 books in 5 different languages and provides a user friendly interface with many features like browsing by facets and full text search. Books are displayed in a specialized book viewer that provides search within the book and hit highlighting. It also provides users, with personalization features like book shelves, text highlighting and comments. DAR Books utilizes DAR publishing API to get the latest books that have been added to the repository. Using the API, DAR books checks the repository at scheduled intervals. The API responds with a list of new books or books that have been updated. DAR books then retrieves the metadata to be indexed in Solr in addition to the new or updated objects to be cached on its servers. This simple yet powerful integration ensures that objects in DAR Books are consistent with their corresponding objects within the repository reducing redundancy and divergence due to storing multiple copies that are not in sync: a problem that faces many applications if they are created as separate silos with no repository integration.

5 The life cycle of a digital object

DAR 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. 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 available for harvesting 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 at the processing stage. 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 next time the application asks the API for updates, it will discover those new items. The application loads both the metadata and the objects from the repository to cache them on its servers. When the metadata of an object changes at organization X, the synchronization plugin loads the new values. The *METS Store* and Fedora are updated and re-indexing of the object is triggered. Whenever application Y checks for updates again using the API, it detects that the metadata of the object is updated so it loads the updated values into its database and displays them.

6 DAR and Scalability

As the number of objects inside the repository increase, several issues arise including the large amount of storage required to store the digital objects, the need to rely on an efficient content model to represent the different types of digital objects and an increase in the number of RDF triples defining the relations among objects.

DAR currently holds more than 450,000 objects including books, photos, manuscripts, maps and documents. These objects occupy around 30TB of distinct items. DAR relies on a storage abstraction layer to isolate the repository from the underlying storage architecture giving us full flexibility to handle the data distribution and load balancing. The storage layer distributes the data among several nodes built from commodity hardware. More nodes can be added as needs arise. Storage nodes are mirrored through rsync to ensure data redundancy. We are currently investigating the usage of rule based storage through iRODS. It provides more flexibility to handle different scenarios based on the usage of objects.

The content model used in DAR provides efficient representation of objects. DAR uses a hybrid atomistic and compound content model providing flexibility of representation. Atomistic content models facilitates a single object's reuse and discovery while at the same time increases the number of RDF triples required for the representation. Compound models on the other side greatly reduce the number of objects and the RDF relations required whenever it is suitable to represent the object in this manner. Books constitute almost half the number of objects in DAR. Books are represented as compound objects thus reducing the number of RDF relations required. 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. Actually, for each book an object is created to hold the content pages, plus an object holding the item data and an aggregate object representing the Bibliographic Record, which might aggregate several items reducing the data redundancy.

DAR heavily relies on RDF triples to store the objects based on its content model, in addition to assigning objects to different sets. An object can be a member of several sets. This results in a large number of RDF triples that poses scalability concerns. The default triple store that comes with Fedora cannot handle the large number of RDF triples efficiently resulting in slow performance for queries. We have investigated several alternatives for scalable triple stores, and successfully integrated 4Store into the repository instead of Mulgara. This involved several modifications: we have written a 4Store driver for Fedora instead of the one that comes for Mulgara, ITQL queries were changed into SPARQL queries and triples were migrated from Mulgara to 4store. We have witnessed a boost in the query response time. As needs arise, 4store can also be installed in a clustered configuration of several 4store nodes to scale even further.

7 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 has undergone several updates since its launch. 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 for the objects was completed in December 2010. BA is currently working on the migration of existing applications to be repository bound thus consolidating

their digital assets and applying the necessary modifications to utilize the publishing API. The potential of Linked Data is yet to be exploited further in DAR and development is underway to enhance the storage layer component by incorporating iRODS in addition to pair trees to manage versioning. We are also working on extending the copyright module to manage more scenarios, and devising solutions to enhance the scalability of DAR.

8 Acknowledgment

Many thanks go to Engy Morsy, Samar Farag, Mohammed Abuouada, Khaled Almahallawy, Niveen Nagy, Yusra Mosallam, Mohamed Mohab, Ahmed Omar and Abdelrahman Mostafa for their efforts in the design and implementation of the system.

9 References

Abrams, S., Kunze, J., Loy, D. (2009) "An emergent micro-services approach to digital curation infrastructure." In the proceedings of The Sixth International Conference on the Preservation of Digital Objects. iPRES 2009 (California: California Digital Library). Available at: <http://escholarship.org/uc/item/5313h6k9>

Apache Lucene (2011). Available at: <http://lucene.apache.org/java/docs/index.html> [Accessed April 19,2011]

Apache Solr (2011). Available at: <http://lucene.apache.org/solr/> [Accessed April 19,2011]

Arrow Project (2008). Available at: <http://arrow.edu.au/> [Accessed August 31,2011]

Burton-West T.(2010a), "Too many words!", Available at: <http://www.hathitrust.org/blogs/large-scale-search/too-many-words>, Accessed [August 31, 2011]

Burton-West T.(2010b), "Too many words again!", Available at: <http://www.hathitrust.org/blogs/large-scale-search/too-many-words-again>, Accessed [August 31, 2011]

Cramer, T., Kott, K. (2010) "Designing and Implementing Second Generation Digital Preservation Services: A Scalable Model for the Stanford Digital Repository". D-Lib Magazine, volume 16, number 9/10. Available at: <http://www.dlib.org/dlib/september10/cramer/09cramer.html>

Davis D., Green R. (2011) "The Hydra Project", Available at: <https://wiki.duraspace.org/display/hydra/The+Hydra+Project> [Accessed August 31, 2011]

Diggory M. (2011) "DSpace Fedora integration". Available at: <https://wiki.duraspace.org/display/DSPACE/Fedora+Integration> [Accessed March 19, 2011]

Dspace (2011). Available at: <http://www.dspace.org/> [Accessed April 20,2011]

DuraSpace (2011). Available at: <http://www.duraspace.org/> [Accessed March 5, 2011]

EPrints (2011). Available at: <http://www.eprints.org/software/> [Accessed April 19,2011]

eSciDoc (2011). Available at: <https://www.escidoc.org/> [Accessed April 19,2011]

Fauduet, L., Peyrard S.: (2010) “A Data-First Preservation Strategy: Data Management In SPAR”. In the proceedings of the Seventh International Conference on Preservation of Digital Objects iPRES, edited by Andreas Rauber (Vienna, Austria: Wien Österreich. Computer Gesellsch). Available at: <http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/fauduet-13.pdf>

Fedora Commons (2011). Available at: <http://fedora-commons.org/> [Accessed March 15, 2011]

Fedora Content Model Architecture CMA (2007). Available at: <http://fedora-commons.org/documentation/3.0b1/userdocs/digitalobjects/cmda.html> [Accessed March 15,2011]

Fedora Performance and Scalability Wiki (2009). Available at: <http://fedora.fiz-karlsruhe.de/docs/> [Accessed august 31, 2011]

Greenstone (2011). Available at: <http://www.greenstone.org/> [Accessed March 15,2011]

Handle system (2011). Available at: <http://www.handle.net/> [Accessed August 31, 2011]

iRODS (2011). Available at: <http://www.irods.org/> [Accessed April 20, 2011]

Metadata Encoding and Transmission Standard METS (2001). Available at: <http://www.loc.gov/standards/mets/> [Accessed April 20, 2011]

Metadata Object Description Schema MODS (2005) . Available at: <http://www.loc.gov/standards/mods/> [Accessed April 20, 2011]

Mikhail, Y., Adly, N., Nagi, M. (2011) “DAR: an eco-system of components for an integrated institutional repository.” The 6th Annual Conference on Open Repositories 2011 (Austin, Texas). Available at: [http://www.bibalex.org/isis/UploadedFiles/Publications/DAR_3_0 - OR 2011_acceptedApril.pdf](http://www.bibalex.org/isis/UploadedFiles/Publications/DAR_3_0_-_OR_2011_acceptedApril.pdf)

Mikhail, Y., Adly, N., Nagi, M. (2011) “DAR: Institutional Repository Integration in Action.” In the proceedings of the International Conference on Theory and Practice of Digital Libraries TPD 2011 and Lecture Notes in Computer Science LCNS, volume 6966, p. 348, (Springer).

Open Archival Information System reference model (2002). Available at: <http://public.ccsds.org/publications/archive/650x0b1.pdf> [Accessed August 15, 2011]

Project Blacklight (2011). Available at: <http://projectblacklight.org/> [Accessed May 12, 2011]

Saleh, I., Adly, N., Nagi, M.(2005) “DAR: A Digital Assets Repository for Library Collections”. In the proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries ECDL, edited by Andreas Rauber, Stavros Christodoulakis, A Min Tjoa. (Berlin ,New York: Springer). Available at: http://www.bibalex.org/isis/UploadedFiles/Publications/DAR_1.pdf

SPAR Project (2011). Available at: http://www.bnf.fr/en/professionals/preservation_spar.html [Accessed April 20, 2011]

Virtuoso Universal Server software (2011). Available at: <http://virtuoso.openlinksw.com/> [Accessed May 13, 2011]

VTLS VITAL software (2011), <http://www.vtls.com/products/vital> [Accessed April 20,2011]

Yakout, M., Adly, N., Nagi, M. (2006) "Digitization Workflow Management System for Massive Digitization Projects." In the proceedings of the 2nd International Conference on Universal Digital Library ICUDL, edited by Ismail Serageldin and Raj Reddy (Alexandria, Egypt: Bibliotheca Alexandrina). Available at:
http://www.bibalex.org/isis/UploadedFiles/Publications/Massive_Digit_Workflow_Mgmt_Sys.pdf

4Store triple store (2011). Available at: <http://www.4store.org/> [Accessed May 20, 2011]