DAR: an eco-system of components for an integrated institutional repository

Bibliotheca Alexandrina (http://www.bibalex.org/)

Introduction

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) 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.

Building on well established standards, DAR is designed with flexibility in mind to integrate with any source of metadata, whether an ILS, repository or database through its plugin architecture. This allows for collaboration with other repositories easily by ingesting some of their collections and synchronizing their metadata through plugins. 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.

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 grouping objects into sets, where an object can belong to different sets allowing administrators to share items among several applications. This enables managing only one instance of the object inside the repository, while not losing the ability to have different derivatives of this same object depending on the application requirements. In managing unique instances of objects, DAR uniquely identifies objects using a persistent handle[1] in addition to heavily relying on RDF relations to define sets. Secondly, DAR provides an API 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 extending its features.

DAR Architecture

Figure 1 shows the conceptual overview of DAR. Objects are consolidated into sets or collections. DAR stores simple derivatives for all objects used for display through the *discovery layer*. The design is OAIS[2] compliant and 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 plugins. A *Digital Assets*

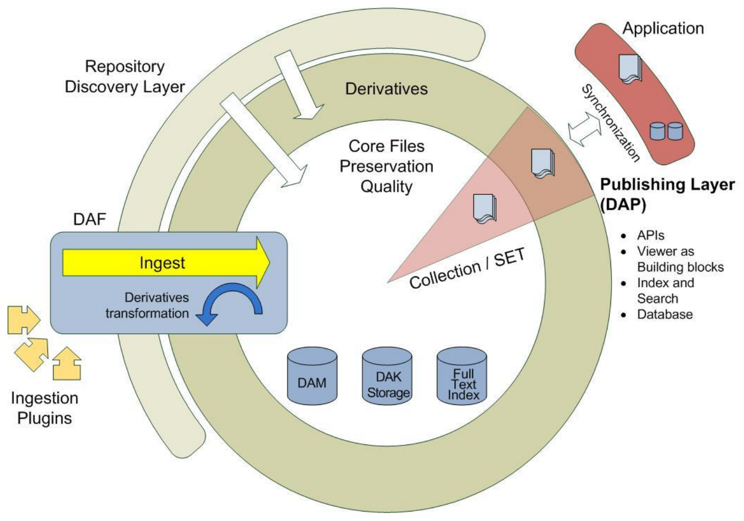

Figure 1: DAR Conceptual overview

[1] The Handle System, http://www.handle.net/
[2] The Open Archival Information System reference model, http://public.ccsds.org/publications/archive/650x0b1.pdf

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.

The modular design of DAR allows for the incorporation of components like Fedora[3], Mulgara[4], the Handle system and Solr[5] to manage metadata in a synergetic way. This modularity is possible through abstractions on the component level.

Figure 2 depicts the architecture of DAR and the following gives an overview of the main components.

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 plugins.

The process of digitization requires the incorporation of several tools, and there is no one-size-fits-all tool that can perform all the digitization tasks. DAF integrates with these tools to assist repository administrators in managing the workflow. It can integrate with automated tools and scripts, checking their status and verifying their output through pre-phase and post-phase checks, making sure the output is compliant to the digitization standards and offloading the human operators to do tasks that humans are good at: e.g. OCR correction.

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. 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. Once ingested, the item will 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).

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. DAR uses a hybrid atomistic and composite content model of the objects providing flexibility of representation. Currently DAR holds more than 430,000 objects including books, photos, manuscripts, maps and documents.

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 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.

[3] Fedora Commons, http://fedora-commons.org/
[4] Mulgara, http://www.mulgara.org/
[5] Solr, http://lucene.apache.org/solr/

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.

A flexible *copyright module* manages who can access the object and what level of access is provided, based on the application requesting access. Levels of access include: e.g. view, download, print, etc. Copyright information is stored with the object. DAM also 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.

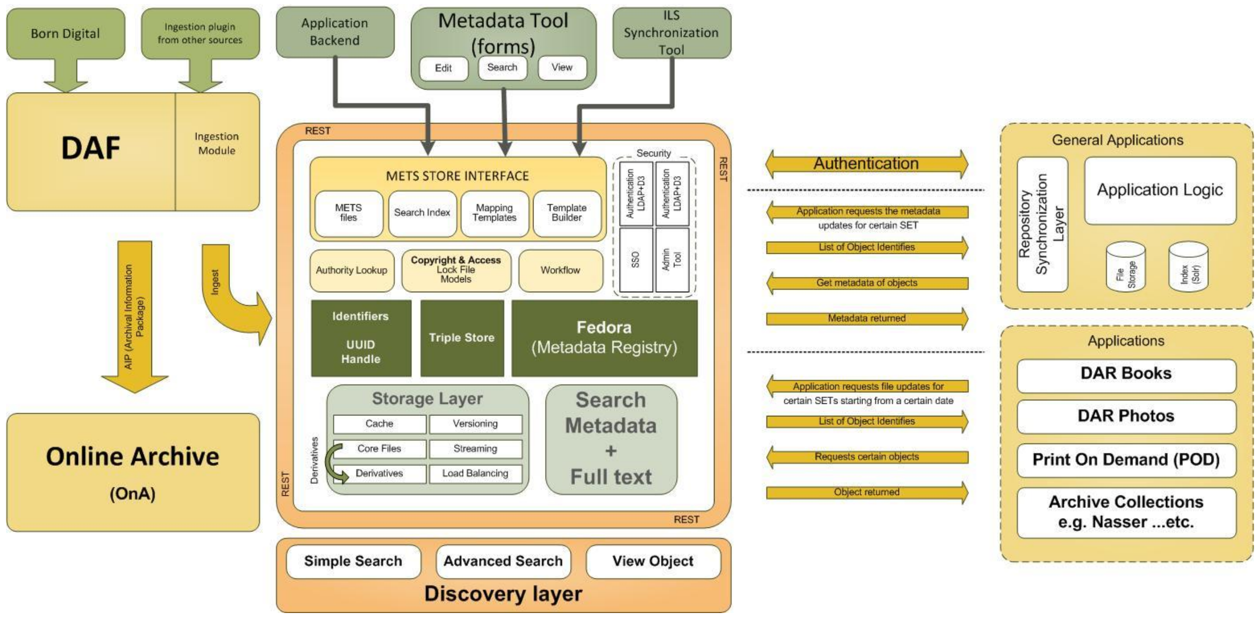

Figure 2: DAR detailed Architecture

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 Digital Assets Keeper (DAK)

The Digital Assets Keeper 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. DAK manages the versioning of items in addition to load balancing across storage nodes. It also handles the caching of derivates used for the repository discovery layer.

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. 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.

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.

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 running special software developed by BA for data management.

Conclusions

DAR is the flagship of Bibliotheca Alexandrina's digital library. 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. Development is underway to enhance the storage layer component in addition to extending the copyright module. The potential of triple stores is yet to be exploited further in DAR. BA is currently working on the migration of existing applications into the repository modifying them to be repository bound thus consolidating the digital assets.