DAITSS Grows Up: Migrating to a Second Generation Preservation System

Priscilla Caplan, Carol C.H. Chou; Florida Center for Library Automation; Gainesville, FL / USA

Abstract

In the early 2000s a number of institutions pioneered the development and implementation of OAIS-based digital preservation repositories. Now with several years of operational experience behind them, some of these institutions are implementing "second generation" repository systems designed to take advantage of lessons learned. This spring the Florida Digital Archive is implementing DAITSS 2, a completely rewritten version of its original DAITSS system. Functionally, DAITSS 2 does not differ greatly from DAITSS, but technically and architecturally there are many changes. Migration to the new system is complicated by the fact that the Florida Digital Archive manages multiple copies of an archival store containing upwards of 30 million files. This paper describes major differences between the two systems, issues in migrating from one to the other, and the final migration plan.

Introduction

In the early 2000s a number of institutions pioneered the development and implementation of OAIS-based digital preservation repositories. Now with several years of operational experience behind them, some of these institutions are implementing "second generation" repository systems, applications designed to take advantage of lessons learned running first generation systems. Some examples include Stanford University moving from the Sanford Digital Repository (SDR) to SDR 2.0 [1]; Cornell University migrating from an aDORe-based system to one based on Fedora; and the State University System of Florida rewriting their DAITSS application as DAITSS 2.

DAITSS is the application that underlies the Florida Digital Archive (FDA), a long-term preservation repository for the use of the libraries of the eleven public universities in the state. DAITSS was written by staff at the Florida Center for Library Automation (FCLA) with some support from the Institute of Museum and Library Services (IMLS), and went into production in November, 2005. It provides automated support for submission, ingest, data management, archival storage, and access (dissemination). It implements preservation strategies based on format transformation, including file normalization and forward migration when objects are ingested and before they are disseminated. DAITSS is described in [2,3].

In 2008 planning began for a complete rewrite of the original DAITSS application (D1) as DAITSS 2 (D2). The initial goals of the effort were: 1) to compartmentalize format support so new formats could be more easily added, 2) to make it simpler to modify existing code and to integrate new code, 3) to make it easier for other institutions to adopt DAITSS and contribute to the code base, 4) to allow multiple service instances to improve performance and scalability, 5) to improve interoperability with other preservation repository applications, and 6) to eliminate complexity that experience had shown to be unnecessary. [4] Later a seventh major goal was added, 7) to improve operational efficiency and make operations data readily available to both FDA operators and affiliates.

Differences between D1 and D2

The first five goals were facilitated by redesigning the sprawling and monolithic D1 as a set of RESTful Web services. In the old architecture, a single program would read each Submission Information Package (SIP) in the "incoming" directory and either reject it or ingest it (store it as an AIP) before reading the next SIP. In the new architecture, packages progress asynchronously through a series of Web services. Each SIP is validated by the Submission Service and if accepted, stored in a workspace as a Workspace Information Package (WIP). The Ingest Process oversees the transformation of the WIP into an AIP. Each file in the WIP is operated on in turn by the Virus Check Service, Description Service, Action Plan Service, Transformation Service, and XML Resolution Service. The finished WIP (now an AIP) is used to populate a fast access database, tarred up into a single file, and handed off to the Storage Service to be stored on a preconfigured set of independent storage pools. Other processes oversee other functions, including Dissemination (access), Withdrawal, Peek, and Refresh. Refresh is an Ingest-like process that takes as input a stored AIP rather than a producer-supplied SIP, and is used to update the contents of the AIP before Dissemination.

Although designed before the term "micro-services" became trendy, D2 provides a good example of the flexibility of a micro-services architecture. Development and testing goes significantly faster, since each Web service can be modified without affecting the others. The individual Web services can be used in standalone ways, or integrated with other applications besides DAITSS. The Description Service, for example, is used by the online PIM (PREMIS-in-METS) application to create PREMIS-conformant descriptions of files. We expect that this and the XML Resolution Service will be widely useful in various contexts.

In addition to the architectural changes, we made an effort to replace local coding with shared tools wherever possible. In D1, format identification, validation and characterization occurred in core code using DAITSS-specific format identifiers and format schema. D2 uses DROID, JHOVE, PRONOM/UDFR format identifiers, and common format schema. Integration of JHOVE2 and FITS is planned for a future release. Using common tools facilitates ease of adoption by other institutions, improves interoperability, and allows other institutions to advantage of work

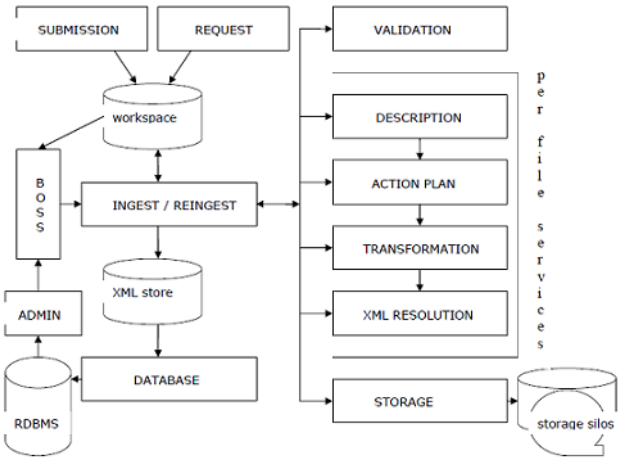

Figure 1. *DAITSS 2 processing diagram*

done for the FDA. D2 developers have added to the DROID format database and contributed various bug fixes to JHOVE.

The first goal was achieved primarily by decoupling the institution's preservation strategy from the codebase. In DAITSS, format specific preservation strategies such as migration and normalization are defined as format action plans. In D1, the action plans are hard coded, so that changing the preservation strategy for any format requires core code changes. In D2, the action plans are defined as standalone XML documents with embedded processing instructions. An action plan can be transformed by a stylesheet for display to users as documentation, or processed by the Action Plan Service to determine the appropriate format transformation instructions which are then passed to the Transformation Service to carry out.

For example, if repository staff decide to normalize all PDFs to PDF/A-1b, all they need to do is write or acquire a converter, add a processing instruction to the PDF action plan, and add the converter API to the configuration of the Transformation Service; no change to the D2 codebase is required.

The fifth goal, improving interoperability, was furthered by the micro-services approach, and also by further standardization. In both D1 and D2, every AIP has an "AIP Descriptor," a METS document that contains all known information about the AIP. This includes not only an itemized list of the other files contained within the AIP, but also all metadata supplied or extracted about the files, all digital provenance recorded by the system, and all relationships among files known to the system. In D1, the extensive metadata recorded within the AIP Descriptor was defined according to locally developed schema intellectually mapped to the PREMIS Data Dictionary. In D2, the metadata is vanilla PREMIS and stored in the AIP Descriptor according to best practices for using PREMIS in METS. [5] Where D1 did not implement, or only partially implemented, many PREMIS elements, D2 implements all applicable elements apart from Environment information. Format-specific technical metadata is recorded using standard schema (e.g. MIX, TextMD, DocMD) where possible and treated as PREMIS extension schema. This makes D2 more standards compliant and facilitates interoperability with other repositories. It also absolves the FDA of the need for extensive local documentation.

The Storage Service decouples the institution's storage strategy from the codebase. In D1, the application could be configured to write AIPs to any number of storage silos, defined as roughly 2TB chunks of storage on any medium. D1 performs each write sequentially, keeping track of the location of each copy in the DAITSS database. The D2 Ingest process knows only the name of the Storage Service, to which it sends the tarred AIP. The Storage Service in turn knows about an arbitrary number of Storage Pool Servers. Each Storage Pool Server has defined to it a set of homogenous storage silos. The Storage Service sees that a copy of each AIP is written to each storage pool, while the Storage Pool Server figures out which silo within the pool should be used in order to make the most efficient use of space. Both the Storage Service and the Storage Pool Servers keep their own databases of information.

The sixth goal of eliminating unnecessary complexity focused on two features of D1. First, D1 allowed affiliates (library users of the FDA) to specify on a file-by-file basis whether they wanted bit-level preservation only, or full preservation treatment including format transformation when possible. Although this sounds straightforward, both conceptually and technically it was fraught with hazard. For example, given that FDA SIPs are supposed to contain complete representations of complex objects (i.e., all the files needed to render the object intelligibly), if some files in a SIP were designated for bit-level treatment and some for full, it would not be possible for the FDA to guarantee the renderability of the representation. Should the code downgrade all files in the SIP to bit-level? What if the affiliate's decisions changed over time, and a file once specified as full is now specified as bit level – should the code provide for deleting migrated versions from the AIP? The affiliates never had a good understanding of how the preservation levels worked, and the code to implement them was suspect. In D2, everything ingested is afforded the most complete preservation treatment available.

The second problematic feature was the D1 handling of files occurring in very many packages, primarily the METS schema, PREMIS schema, and local schema required for AIP Descriptors. In D1, these files were pre-identified as "global files" and stored only once in special packages. During Ingest, if any of these files were present in a SIP, the file would be replaced in the AIP by a reference to the stored global file. In theory, when the AIP was transformed in to a DIP for dissemination, the global files would be retrieved and inserted in place. At the time D1 was designed this seemed like a better idea than storing redundant copies of the same files in hundreds of thousands of packages. However, the code for handling global files was enormously complex and subject to errors, which led in some cases to packages that could not be properly disseminated. In D2, the XML Resolution Service downloads all the schema needed to validate the files in the SIP and tars the schema up into a single file (informally called a "schemaball") which is included in the AIP.

The final goal of improving operations information rectifies a major omission in D1. In D1, an identifier is assigned to a package and information about the package is recorded in the DAITSS database only when the AIP is written to storage. A package could be rejected at any point during ingest processing,

because of either bad data or program bugs. A rejection report would be emailed to the submitter, but there would be no stored information in the system about the existence of that package or the reason for rejection. Over time, standalone tools were written to supplement D1 information. The PackageTracker tool provided a suite of scripts allowing operators to register each incoming SIP in an operations database and record every change of its location until ingested. A RejectsTracker tool extracted information from the rejection report and inserted it into a second operations database. Neither database was known to the D1 application code.

In D2, every package submitted to the Submission Service is assigned a unique identifier at the point of submission, and tracked in the DAITSS database from the start. The only cause for rejection is if the package fails validation by the Submission Service, in which case details about the validation failure are recorded. If a package passes validation, it has good data by definition, and cannot be rejected at any later point, only "snafued". The basic difference between a reject and a snafu is that a rejected package is not stored in the system and must be resubmitted to be processed; a snafued package is stored as a WIP and will ultimately continue through processing when the problem preventing progress is fixed. As Ingest progresses the WIP is updated with new information provided by each Web service in turn, which is ultimately added to the database when the AIP is stored. Therefore D2 has no need of external tracking databases; all information is recorded in the DAITSS database or present in the in-process WIP.

Inventory of changes

Changes affecting the data store:

Both D1 and D2 store a copy of the AIP Descriptor in the stored AIP itself, so that each AIP in storage is fully self-describing and does not depend on any computer application or external database to be understandable. Therefore any change to the AIP Descriptor necessitates a change to the data store:

- The AIP Descriptor stored in each package must be changed to reflect the move from local to standard schema.
- The information in the AIP Descriptor for each file in the package must be regenerated as each file is re-identified, assigned a PRONOM/UDFR format identifier, and described according to a standard schema.
- The AIP Descriptor stored in each package must be changed to remove any references to the D1 global files.
- The schema stored in D1 as shared global files must be found, aggregated into a schemaball and copied into the AIP.

Changes affecting the DAITSS Database:

Both D1 and D2 implement a relational database which redundantly stores selected metadata from the AIP Descriptor for fast access. The D2 database also includes additional information not included in the D1 database.

- The D2 database table schema reflects the changes made to the AIP Descriptor.
- In D2, a copy of the entire AIP Descriptor is stored in the database as a blob. This is an experiment we might reverse, but was done on the theory that most questions can be answered from information in the AIP Descriptor, obviating the need to retrieve packages from Archival Storage.
- The D2 database includes operational information about rejected packages and packages in transit which in D1 is stored only in external databases. Information must be transferred from the external databases and reformatted to populate the D2 DAITSS database with historical information.
- Changes to storage handling means that storage location data in the D1 database is now stored in a separate database used by the Storage Service, and filesystem information is stored in new databases used by the Storage Pool Services.
- D1 uses MySQL but D2 is designed to be database agnostic, and has been tested with MySQL and PostgreSQL. The FDA made the decision to implement D2 with PostgreSQL.

Migration

The goals of the D1 to D2 migration are 1) to convert all D1 packages into D2 packages, and 2) to make the D2 database complete and accurate for the entire archive. Constraints over the migration process are 1) that FDA processing cannot be down for more than a week, and 2) that operations staff need to have all reporting information about the location and status of packages immediately available to them.

We considered two different approaches to the migration. The first approach would treat the conversion of the DAITSS databases and archival store as an independent process external to the DAITSS application. Programs could be written to read each stored AIP directly, create a schemaball, reformat the AIP descriptor, and populate the databases with D2 information. Alternatively, we could handle the migration as a DAITSS process equivalent to a giant Refresh of the datastore. The first approach would be faster and more direct, but the second was preferred as more in keeping with the philosophy behind both D1 and D2, which is that all actions take place within the system where digital provenance is strictly maintained. It also had the advantage that code already written for D2 could be used with only a small amount of additional programming.

We estimated that using the Refresh approach, it would take somewhere between 6 and 18 months to convert the entire D1 data store to D2 format, making it obvious that migration and operations were going to have to proceed simultaneously. Several strategies for structuring the migration were then considered, ranging from a migration-on-demand approach that would leave packages untouched in D1 until requested by an affiliate, to a mass dissemination from D1 to be ingested into D2. Since FCLA is a partner in the TIPR (Towards Interoperable Preservation Repositories) project, we gave particular consideration to exporting AIPs from D1 as Repository Exchange Packages (RXPs), and importing the RXPs into D2. The RXP was designed with system migration as one use case, and allows package-level provenance to be communicated as well as the package structure. [6] The Refresh approach, however, allowed us to recreate and

rewrite each package without going through an actual Dissemination, so using an RXP was unnecessary.

The plan ultimately implemented is a hybrid that converts some database elements immediately but leaves other data and stored packages to be migrated over time. Migration will proceed in three phases:

1) Populate the D2 database with the minimum amount of information required by the system to run and by operations for reporting. This includes all data about affiliate user accounts; abbreviated information about each package (including complete provenance information) from the D1, tracker and rejects databases; and abbreviated information about each file in the package from the D1 database. Create an abbreviated version of the new AIP descriptor (called a "stub") and store as a blob. At the same time, populate the D2 Storage Server and Storage Pool Service databases from the D1 database.

2) Stop running D1. Put D2 into production. Wait a reasonable period of time to work through new procedures, bugs and bottlenecks.

3) Begin an ongoing conversion of D1 AIPs to D2 AIP format. This will be done by a special version of the Refresh process. For each AIP, Refresh will retrieve the package from D1 storage and extract all (and only) files originally submitted by the affiliate. For any original files requiring external schema for validation, Refresh will retrieve the appropriate D1 global files and tar them up into a schemaball. It will then process all files through normal D2 processing, creating transformed versions as necessary and building the new AIP as it goes along. Also as normal D2 processing, the new AIP Descriptor will be constructed and used to update the fast access database. Finally, the AIP is written to the D2 data store and the old AIP is deleted from D1 storage. D1 storage can be recycled for use by D2 silo by silo as space is freed up.

The ongoing conversion can run concurrently with normal D2 Ingest processing. If a D1 package is requested for Dissemination before it has been converted, a DAITSS operator can force an immediate Refresh on the AIP.

Conclusion

Migrating from a first-generation to second-generation preservation repository system is not for the faint of heart. The Florida Center for Library Automation, which develops and runs the Florida Digital Archive on behalf of the State University System, is convinced that the DAITSS 2 development project was well worth the investment. The new system will be far easier to operate and to enhance, it is easier to understand and to document, and it will achieve better throughput and performance. We hope that other institutions or consortia with the capacity to run an OAIS-based preservation repository will be able to implement the DAITSS application with relative ease, and customize it to reflect their own preservation strategy decisions.

Designing, developing, testing and implementing the new system was a major investment in our digital future. Nonetheless, with nearly 300,000 stored AIPs comprising more than 30 million files, the launch of DAITSS 2 will be followed by many months of data migration. Our decision to Refresh (recreate) each package means a longer migration time than some alternative approaches, but allows a fairly organic transformation in keeping with the preservation philosophy of the repository. We hope it will also give us some insight into the time and resources a mass file format migration might require in the future.

References

[1] Tom Cramer, Katharine Kott, "Designing and Implementing Second Generation Digital Preservation Services: A Scalable Model for the Stanford Digital Repository," D-Lib Mag, 16 (2010)

[2] Priscilla Caplan, "The Florida Digital Archive and DAITSS: a model for digital preservation", Library Hi Tech, 28, 224 – 234 (2010)

[3] Priscilla Caplan. "The Florida Digital Archive and DAITSS: a working preservation repository based on format migration." Int. J. Digit. Libr. 6, 305-311 (2007)

[4] Carol Chou, Randall Fischer, Franco Lazzarino. "Updating DAITSS: Transitioning to a Web Service Architecture." iPRES Procedings 2008. http://www.bl.uk/ipres2008/presentations_day2/35_Fischer.pdf

[5] Guidelines for Using PREMIS with METS for Exchange. http://www.loc.gov/standards/premis/guidelines-premismets.pdf

[6] Priscilla Caplan, William Kehoe, Joseph Pawletko, "Towards Interoperable Preservation Repositories: TIPR" Int. J. Digit Cur. 5 (2010)

.

Author Biography

Priscilla Caplan is the Assistant Director for Digital Library Services at the Florida Center for Library Automation, where she oversees the Florida Digital Archive. Carol C.H. Chou is a lead developer for DAITSS and Formats Specialist for the Florida Digital Archive.