Designing a Migration Path

Final Report

September 2019

INSTITUTE of Museum and Library SERVICES

This project was made possible in part by the Institute of Museum and Library Services grant number LG-72-18-0204-18.

DURASPACE™

Designing a Migration Path Project Co-Directors and Report Authors

David Wilcox
Product Manager
DuraSpace

Erin Tripp
Executive Director
DuraSpace

Designing a Migration Path Advisory Board

Sayeed Choudhury
Associate Dean for Research Data Management
Johns Hopkins University

Mike Giarlo
Digital Library Software Engineer and Architect
Stanford University

Mark Jordan
Associate Dean of Libraries, Digital Strategy
Simon Fraser University

Este Pope
Head of Digital Programs
Amherst College

Scott Prater
Digital Library Analyst
University of Wisconsin-Madison

Tim Shearer
Associate University Librarian for Digital Strategies and IT
University of North Carolina at Chapel Hill

Andrew Weidner
Digital Operations Coordinator
University of Houston

Table of Contents

Executive Summary

Hundreds of libraries and archives in the United States and abroad use Fedora repository software to deliver scholarly, scientific, and cultural heritage resources and services to patrons. Fedora 4 was released in 2015, followed by Fedora 5 in 2018, but most of the community is still using Fedora 3. Fedora 4 is a completely re-written application that shares no code with Fedora 3. Fedora 5 is the first release to use semantic versioning: it is built on the same codebase as Fedora 4, but a small set of API changes necessitated an incremental change to the major version number. Currently, both Fedora 4 and Fedora 5 are officially supported releases, while Fedora 3 is no longer being supported.

In 2018, DuraSpace was awarded a planning grant (LG-72-18-0204) by the Institute of Museum and Library Services (IMLS) to investigate the barriers to upgrading libraries and archives running unsupported versions of Fedora because reliance on the older versions of Fedora puts the stability, security and functionality of the content and services of these repositories at risk. The grant team conducted background research, including an environmental scan, a collection of institutional profiles, and assessments of relevant technologies, before drafting and distributing a survey to the international community. The survey closed with 111 responses, which have been analyzed and summarized in this report. A select group of survey respondents were also invited to participate in a focus group to discuss some of their responses in more detail.

The key finding of this project is that any software upgrade (including a migration) that requires significant effort creates a decision point. The institution must decide whether to go forward with the upgrade, stay on the current version, or move to another platform, and the answer all depends on value. The new version of the software needs to provide enough additional value compared to both the current version and other software applications to justify the cost of the upgrade. Many survey and focus group respondents said they did not see enough additional value in Fedora 4.x or 5.x to justify the level of effort required to upgrade from Fedora 3.x. Therefore, the recommendations focus on increasing the value of Fedora while decreasing the level of effort required to upgrade:

1. Develop and deliver Fedora 6 with support for transparent persistence (preservation).
2. Develop migration tools to support upgrades from Fedora 3, 4, and 5 to Fedora 6.
3. Test Fedora 6 at scale under a variety of deployment scenarios, including cloud-based deployments.
4. Work with pilot institutions to document case studies, data model examples, and best practices for each community moving to Fedora 6.
5. Communicate the need to upgrade from Fedora 3.x to avoid stability and security issues.
6. Communicate the value of migrating to Fedora 6 in several contexts:

    a. As an open source, community supported project.
    b. As insurance against vendor/hardware lock-in.
    c. As part of a portfolio of open source programs supported by LYRASIS.
7. Pilot a repository upgrade service through LYRASIS.

These recommendations provide meaningful steps that could be taken to lower the barriers to major repository upgrades, both in the specific case of Fedora and for repositories more generally.

Environmental Scan Summary

Repository upgrades and migrations are quite common, and the literature covers several important aspects of this process: motivations for undertaking a migration, the difficulty of migrations, the possible benefits of a migration, and advice for those looking to undertake a migration in the future.

A common motivation for repository migrations is the cost of a commercially licensed product. Gilbert and Mobley [8] were facing an increased cost to their CONTENTdm license due to reaching the item limit of their current tier, and Stein and Thompson [18] cited license and maintenance fees as one of the main drivers of repository migrations based on survey data. Issues with the commercial platform itself, from performance and scale limitations (Neatrour et al [11], Witkowski et al [25]) to a lack of flexibility with regard to file and metadata formats (Gilbert and Mobley [8], Wu et al [26]), were also key motivators. Finally, better support for digital preservation (Stein and Thompson [18], Berghaus et al [2], Fallaw et al [6]) and linked data (Wu et al [26], Stein and Thompson [18]) rounded out the top motivators in the literature.

There are many factors that make migrations difficult, but there is one primary problem category throughout the literature: metadata. Van Tuyl et al [22] cite metadata remediation as the biggest time sink during their migration project, and many others (Bridge2Hyku Team [3], Gilbert and Mobley [8], Neatrour et al [11]) present case studies that involve significant time spent on metadata normalization, deduplication, and remediation. This speaks to a related difficulty often cited in the literature: inconsistent or “messy” source data. The process of mapping metadata from one repository system to another would be much simpler were it not for the fact that many legacy systems tend to have metadata quality problems in the form of custom local fields, duplicate fields, and misspelled entries.

There is a great deal of migration advice to be found in the literature, based primarily on lessons learned from migration projects. Tripp [20][21] summarizes much of this advice into four categories: planning, metadata normalization, migration, and verification. Each of these categories is represented in the rest of the literature; Nowak et al [12] undertook a great deal of planning for their migration project, while Simic and Seymore [16] invested a lot of time in large scale metadata normalization prior to migration. The migration phase itself was often accomplished with a combination of scripts and manual intervention, and the same is true of the verification step.

Common Themes

- Motivations for migration
    - Commercial license costs and limits
    - Lack of flexibility
    - Staff investment vs. licensing fees
    - Performance and scale issues
    - Better integration with other applications/services
    - Support for linked data
    - Support for digital preservation
- Migration difficulty
    - Custom metadata fields
    - Inconsistent data
    - Different data models
    - Metadata mapping: e.g. MODS XML to RDF
    - OSS documentation is not always complete/accurate
- Migration benefits
    - Metadata improvement/enrichment
    - Skills development
    - Streamlined workflows
    - Enhanced discovery via metadata enrichment
- Migration advice
    - Communicate
        - Engage with stakeholders, collect feedback, report on progress
    - Work with a representative sample
    - Clearly define requirements, scope
    - Normalize metadata before migration
        - But carefully scope this effort
    - Iterate, spot check
    - Use Agile methodologies
    - Plan for contingencies: staff turnover, learning curve, avoid single points of failure
    - Identify clear roles and responsibilities
- Repository requirements
    - Support flexible object types and metadata
    - Support batch ingest (e.g. from a spreadsheet)
    - Large community
    - Modular
- Status of Fedora
    - Most still using Fedora 3
    - Plans to migrate but few timelines
    - Samvera/Fedora has major performance issues
    - Is the value of Fedora worth the complexity it introduces in the Samvera stack?

- Tools
    - There are several examples of tools developed to aid/automate migration activities

Fedora API Review Introduction

Since the release of Fedora 4.0, Fedora has provided a robust REST-API based primarily on the Linked Data Platform (LDP) as the pattern for resource management operations (Create, Read, Update, and Delete; or CRUD). Over time, the other services provided by the Fedora API (versoning, authorization, fixity, and messaging) have been aligned with other relevant web standards. However, these services and standards were subject to change, and the API was not versioned separately from the underlying implementation, so there was no clear way to distinguish between Fedora API changes and Fedora software updates without reading detailed release notes. The API specification changes this paradigm such that the services Fedora offers and the standards with which it complies are now formally documented and versioned separately from the underlying application. This provides stability and predictability for clients, while also unlocking the potential to replace the underlying application with a different implementation that satisfies different use cases. This approach may also provide benefits for repository migrations because tools can be written against a particular version of the API, which will change less often than the software application.

The Fedora API

The Fedora API is primarily an extension of the LDP 1.0 specification, with the following enhancements:

- reconciliation of [LDP] and the version identification and navigation scheme delineated in the Memento specification [RFC7089],
- integration with the Web Access Control proposal [SOLIDWEBAC],
- design for the publication of event notifications, and
- interaction patterns to support binary resource fixity verification.

The primary goal of the API specification is to clearly define the behaviours of Fedora implementations to promote stability and interoperability with client applications. A client should be able to be developed against the specification without knowing anything about the underlying implementation, and that same client should work with a different Fedora API implementation of the same version without making any major changes.

The main components of the API specification are as follows:

Resource Management - Linked Data Platform (LDP)

LDP defines interaction patterns for clients and servers on the web using linked data (RDF). LDP defines two primary types of entities: containers (composed of RDF) and binaries (not

composed of RDF; essentially files). Fedora supports these entity types, along with some more specific types of containers, and uses these concepts for resource management. Everything a user creates, reads, updates, or deletes in a Fedora repository is either a container or a binary; containers are used to represent intellectual objects and binaries are typically files uploaded from a desktop of server (PDFs, images, videos, XML metadata files, etc.). The native response format for requests made against these resources is RDF.

Versioning - Memento

Fedora supports versioning for both containers and binaries; a new version can be created as part of any resource management operation, and previous versions can be restored. These interactions are provided in accordance with the Memento specification. Memento provides a mechanism for navigating previous versions of a web resource using standard HTTP operations on the web. A resource that supports Memento provides an HTTP link to a TimeGate resource that manages all the previous versions (called Mementos) of that resource. A client can issue a request (including a date and time) to the TimeGate, which will respond with a link to a Memento that most closely matches the date and time of the request. Thus, a client can issue a request for a Memento of a Fedora resource and get an appropriate response without knowing anything about the underlying Fedora software implementation.

Authorization - Web Access Control

Fedora provides authorization but not authentication. Authentication systems, which validate a user's credentials (typically a username and password) then forward you to an application once the credentials have been verified, are assumed to operate at a layer above the Fedora API. Once a user has been authenticated, Fedora provides authorization support: the user will have different levels of access (read, write, delete) for different resources (or groups of resources) depending on what policies have been set. The authorization policies are based on Web Access Control, a standard for creating and managing policies using RDF. Administrators can control access to repository resources by creating and managing these policies.

Notifications - Activity Streams

Fedora emits notifications after every durable change to a repository resource. Any create, update, or delete operation is considered a durable change. These notifications are messages constructed in accordance with the Activity Streams 2.0 standard. While Fedora doesn't perform any further operations on these notifications after they are sent, the notifications can be used to power external integrations. These integrations are fundamentally asynchronous, and can be used to perform useful background operations such as indexing for search, generating derivatives for binary resources, and triggering exports to other applications and services.

Fixity - HTTP Headers

Fedora supports binary resource fixity in two ways: transmission fixity and persistence fixity. For transmission fixity, a checksum may be included with a binary file when it is uploaded to Fedora. In this case, Fedora will calculate the checksum of the file and reject the upload if there is a mismatch. If the checksums match, Fedora will store the file along with its checksum as an RDF

property. For persistence fixity, a client can request that a checksum of a binary resource stored in Fedora be calculated and returned at any time. This checksum can then be compared against a known value (e.g. the checksum stored as a property on the binary resource).

Impact on Migration

The Fedora API specification can support a path to migration because migration tools can be built against the API. For example, a tool could take exported data from Fedora 3.x, convert it to a format compatible with Fedora 4.x and higher, and import the data via the Fedora API. This has the advantage of providing a stable interface to build tools against; the API will change more slowly than the underlying application, so tools that work against the API will continue to be useful as Fedora undergoes application upgrades over time. However, there are still some complications to be worked out in terms of mapping access controls from Fedora 3.x to the new Web Access Control standard used by Fedora 4.x and higher. Versions and fixity information stored in Fedora 3.x will also need to be addressed as part of the migration.

The primary disadvantage to API-based migrations is performance. Transferring large amounts of data over HTTP can be slow, and the REST API performs a number of operations during object creation, particularly for objects with many internal relationships, that can create an ingest bottleneck. There are ways to mitigate these disadvantages, however, and they may not be relevant for institutions with relatively small collections or longer timelines to complete a migration. One mitigation strategy is to convert XML metadata to RDF properties, thus cutting down on the number of individual files that need to be created.

Another disadvantage is the nature of application upgrades over time. The application underlying the Fedora API is likely to change more quickly than the API itself, and the API specification does nothing to mitigate these changes over time. While the API specification does make migrations easier from a tooling perspective, the difficulties of moving large amounts of data will only become greater as the amount of data managed by an institution grows. One possible mitigation strategy would be to leverage Fedora's external content capabilities to store some or all binary content outside Fedora's native filesystem on some other system that is unlikely to change very quickly.

Fedora API Review Conclusion

The Fedora API specification describes the services Fedora provides in alignment with a set of modern web standards. The specification represents a stable, independently versioned interface on top of the Fedora software application that offers stability for clients and flexibility for alternate implementations that service different use cases. In terms of migrations, tools can be built against the Fedora API to support data imports from previous versions of Fedora, and going forward the API specification will make it much easier to support API-based migrations in future versions of Fedora. While there are disadvantages in terms of performance and future

application changes, the API offers a clear benefit to anyone pursuing a migration to a current version of Fedora.

Oxford Common File Layout Introduction

The previously mentioned disadvantages of the Fedora API may be mitigated by a related community initiative called the Oxford Common File Layout (OCFL). The OCFL is a community-based effort to develop “an application-independent approach to the storage of digital information in a structured, transparent, and predictable manner. It is designed to promote long-term object management best practices within digital repositories.” The original idea for this effort came from a Fedora Camp hosted at the University of Oxford in September 2017, so it has roots in the Fedora community even though the OCFL is not dependent on Fedora in any way. However, if Fedora were to store repository resources as specified by the OCFL, major upgrades in the future may become much easier. One of the primary motivating use cases of the OCFL is to reduce the need for data migrations; both because moving large amounts of data is time-consuming and because such migrations risk data corruption and loss. The OCFL decouples the structure of files on disk from the application(s) that manage the files by defining a file and folder hierarchy to which applications must conform. Thus, the files can be left in place and new software applications can implement the OCFL specification in order to manage the files.

The OCFL Specification

An OCFL Object is “a group of one or more content files and administrative information, that are together identified by a URI. The object may contain a sequence of versions of the files that represent the evolution of the object's contents.” An OCFL object should contain all the data and metadata necessary to represent the intellectual entity and, in the case of disaster, rebuild it completely simply by reading the files on disk. An OCFL object has a specific structure that applications must conform to:

```
[object_root]
  ├── 0=ocfl_object_1.0
  ├── inventory.json
  ├── inventory.json.sha512
  └── v1
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        └── ... content files ...
```

This structure includes a file announcing the object's conformance with a particular version of the OCFL, a directory for each version of the object, and an inventory of the object's contents, including checksums.

Impact on Fedora and Migrations

There are two primary ways the OCFL could be implemented in Fedora: 1) as the structure of a native persistence layer beneath the Fedora API or 2) as an exported backup of the repository that could be used to rebuild the repository in the event of a disaster. Both approaches will be described and analysed below.

Implementing the OCFL in the Fedora Persistence Layer

The recently specified Fedora API separates the services Fedora offers from the underlying software that stores and preserves the contents of the repository. This provides an opportunity for the Fedora community to design and develop a software application that natively stores and manages the contents of a Fedora repository as OCFL objects. The OCFL specification is unlikely to go through major revisions, so repository data would be able to remain largely unchanged over time as new versions of Fedora are deployed over the files and folders on disk.

This approach has several advantages: primarily, it eliminates or at least substantially mitigates the need for data migrations, as the data can remain at rest on a storage platform and OCFL-aware applications can be deployed over them. As repositories manage greater and greater amounts of data, migrations become more costly and time consuming. While migrations might still be necessary in the case of storage hardware upgrades or failures, the cycle should be slower compared to standard repository upgrades and migrations. Additionally, it would be possible to completely rebuild the repository from the data on disk in the case of a disaster; currently this is not possible because the metadata is stored in a separate database. Finally, repository contents would be both machine and human readable, which is a high priority for preservation specialists.

The primary disadvantage to this approach is performance and scale. File systems are reliable but slow compared to databases and other modern storage technologies. Reading and writing to disk is a relatively slow operation, and one that is difficult to scale. However, with the file system model, performance tuning can be effectively pushed down to the operating system and hardware layer, rather than needing to be dealt with in the application. Performance issues can also be mitigated via a modular application design that uses fast technologies such as databases and caches to power repository access while a slower, asynchronous operation writes the data to disk as OCFL objects in the background. This approach would preserve the benefits of scalable and performant technologies while adding the value of an OCFL-conformant storage layer.

OCFL as a Fedora Export

The second implementation possibility is an export tool that would transform natively stored Fedora data into OCFL objects and export the contents to an external location. A manual or automatic operation could execute the export at appropriate times and the exported files could be used to update an existing export such that it exactly mirrors the contents of the live repository.

This approach has all the same advantages around rebuilding the repository from files on disk and providing machine and human readability, and it would be easier to implement because it doesn't require any changes to the native repository storage layer. An Import/Export Utility also already exists, and this tool could be modified to support OCFL export.

Unfortunately, this approach would have no impact on migrations. The native repository storage layer would remain unchanged, so the data would still need to be migrated in the case of a major repository upgrade.

Oxford Common File Layout (OCFL) Conclusion

The Oxford Common File Layout represents a great opportunity for the open source preservation community to standardize on a shared approach to structural file storage. Fedora could benefit from this initiative in several ways, most notably by offering repository rebuild capabilities and self-describing, human-readable persistence. In terms of migrations, the greatest benefit can be achieved by implementing an OCFL-compliant application under the Fedora API. While this approach has several design challenges to overcome, and would take significant community resource investment, it has the potential to make future repository migrations far less onerous, and provide a stable storage specification for Fedora and other repository systems that can last for decades.

Institutional Profile Summaries

We profiled eight institutions using different front-end applications (Islandora, Samvera, custom) supporting different use cases (institutional repository, special collections, research data, etc.), of differing institution and repository sizes. These profiles were used to establish commonalities and differences between Fedora 3.x repositories in order to assess the impact on migrating to the latest version of Fedora. The results are summarized below, broken down by front-end application category. The eight institutions are:

- Florida State University
- National Library of Medicine
- University of Wisconsin-Madison
- UNC Chapel Hill
- Michigan State University
- Stanford University

- Williams College
- Amherst College

Front-end Application

Islandora

Islandora is a Drupal-based framework that interacts with Fedora 3.x largely through the REST-API. Islandora takes a modular approach, consisting of a single core application that can be customized by enabling and configuring Drupal modules. This approach makes migrations easier because the majority of the community is using a common core application with similar data models. Islandora uses a Solution Pack framework; each Solution Pack handles a particular type of content and describes specific data models which are shared across the community. The profiled institutions use a common and relatively small set of file formats, most of which are already supported in Islandora 8. Migrations will also be made easier by using the Drupal Migrate ecosystem, which allows administrators to migrate from an Islandora 7 application to an Islandora 8 application based on Fedora 5.x. Such a migration is necessary because Islandora 7 only supports Fedora 3.x, while Islandora 8 only supports Fedora 5.x and higher.

Islandora 8 continues to follow the same model of providing a core application with configurable modules, which mitigates the need for adopters to rebuild all of their Islandora interfaces and workflow tools - many of these will be part of the standard Islandora 8 application stack. However, some institutions have customized their interfaces, and these customizations will need to be refactored in Islandora 8, which may be challenging for institutions with limited resources. Islandora 7 has also been around for many years and it has accumulated many modules and features which have not yet been built into Islandora 8. This will present a barrier to anyone using Islandora 7 features that don't exist in Islandora 8 yet. Fortunately, as community members build these features they will be available for everyone to use.

Each of the profiled institutions with Islandora repositories have made some front-end customizations to their repositories, and these customizations are largely unavailable to the public and may or may not be documented. This presents a challenge for migrations since these local customizations are unlikely to be developed as part of the core Islandora 8 offering, thus requiring some resource investment on the part of the affected institutions in terms of updating the code to work with Islandora 8[1].

Samvera

The Samvera (formerly Hydra) community takes a different approach from Islandora, with an ecosystem of many different applications and tools used by different institutions. There is no single Samvera application stack, but most applications and tools are written in the Ruby

[1] Islandora has a support timeline based on the Drupal support timeline. Support for Islandora 7 will be phased out by 2022.

programming language and Rails web framework. The most commonly used front-end application (Hyrax) has already been made compatible with Fedora 4.x and 5.x; in fact, the Samvera community was an early adopter of Fedora 4.x, and a previous version of Hyrax (Sufia) was the first application based on Fedora 4.x to be used in production. This has allowed some institutions to migrate to the new platform, though the diversity of community deployments and prevalence of local customizations makes migrations more challenging.

Stanford is an example of a founding Samvera community member that uses a diversity of applications and tools in a complex repository ecosystem. Many of these tools are customized to work within the local environment in particular ways, which makes a repository migration a daunting prospect. The level of customization and interdependency means that Stanford would likely need to undertake a migration of their application framework in-house using local resources; no one else in the community has a similar enough system to work together on a migration.

Custom

Custom Fedora 3.x implementations are the most difficult to profile because each one is different. The institutions we profiled tended to support similar services (ingest, search, administration, dissemination, etc.) but they provide these services in different ways using different applications and programming languages. In some cases these frameworks are available on GitHub, but most often they are managed in-house. Even when the source code is publicly available, it is often not well-documented and not designed for use outside the home institution.

Custom frameworks present a great challenge for migrations because they necessarily require local effort that can't be supported by others in the community. One of the profiled institutions has nearly completed a migration to the latest version of Fedora but it took a lot of effort over a period of several years. Local resources are therefore a potential barrier to migration in the case of repositories with custom front-ends, but as the data model analysis section below describes, shared community tools may be able to support the migration of the data independent of the front-end applications.

Data Model Analysis

Each of the eight profiled institutions provided representative sample data which was used to compare and contrast the data models used by each institution. The models themselves can be found in the Data Models section; what follows is a summary of the findings.

There are many commonalities among the types of data managed by each institution. For example, all eight institutions manage image, document (PDF), video, and paged content (books, serials) items. These objects tend to be modeled similarly across institutions, with the highest degree of consistency between the Islandora repositories. However, there is also some variance, as we see with the University of Wisconsin-Madison: each object is contained by a parent, metadata-only object, which adds an extra layer of hierarchy that does not exist in the

other profiled repositories. Digging into the details further reveals a number of differences between institutional data models.

Each Fedora 3.x object is composed of a small amount of administrative metadata and datastreams, which are files associated with the object, each of which has an identifier. A few datastreams are internal Fedora datastreams, which control how objects are defined and related to each other, permissions, and auditing information; the rest are user-defined content datastreams. In addition, there are some configurable Fedora-specific objects, which are used to structure and define user objects. A migration plan should take into account the information contained in the Fedora 3.x administrative objects, all object administrative metadata, Fedora-specific object datastreams, datastream administrative metadata, and the user-defined datastreams.

User-defined datastream IDs are configurable, and we see a high degree of variance between the datastream IDs used at each institution. In some cases datastream IDs are employed inconsistently between data models within the same institution. This presents a possible barrier to migration because shared tools will need to support custom datastream IDs, either through configuration or local customization. That being said, it would be possible to simply migrate each datastream as a file in Fedora 4.x or higher, thereby maintaining the basic structure of each object.

We also see differences in the focus of each repository. For example, UNC Chapel Hill and Stanford each make heavy use of PREMIS preservation metadata, while the other institutions do not include PREMIS metadata in their data models. Similarly, the National Library of Medicine and the University of Madison-Wisconsin maintain METS structural metadata, while the others do not. These differences are less important from a simple file migration scenario as described in the previous paragraph, but they could present challenges in cases where, for example, PREMIS metadata is transformed from XML to RDF during a migration.

Data transformations are challenging because it is not always clear how the data should be transformed. Many of the profiled institutions use MODS descriptive metadata in Fedora 3.x, and it would be fairly simple to migrate these datastreams and store them as XML files in Fedora 4.x and higher. This is likely to be a sufficient migration for some institutions; however, others will want to take the opportunity to more fully embrace linked data by transforming XML-based metadata into RDF properties. But the transformation from MODS XML to MODS RDF is not a simple one - a Samvera working group only recently published recommendations on how to handle the mappings, and this was a multi-year, multi-group effort. Even in this case, not all institutions will agree on the mappings, so transformation tools need to be robust and configurable.

Migration Tool Review

Since the release of Fedora 4.0 in 2015, several community-supported migration tools have been developed. Each of these tools takes a different approach and serves a limited set of use

cases. The tools focus on data migrations only - they do not provide assistance with updates to front-end applications. The tools include:

- migration-utils
- FedoraMigrate
- Migrate_7x_claw (based on The Drupal Migrate framework)

What follows is an assessment of each tool with a particular focus on current migration use cases.

migration-utils

First developed in early 2015, migration-utils is a Java-based command-line utility that iterates over the FOXML resources in a Fedora 3.x repository (either within the native filesystem or as exported content) and transforms them into Fedora 4-compatible resources before ingesting them into a Fedora 4.x repository. Any versions that were created in the Fedora 3.x repository can also be transformed and ingested into a Fedora 4.x repository. A Spring XML configuration file is used to define the source and destination repositories, as well as the nature of the Fedora 3.x content.

This utility uses a default set of property mappings that can be found in the README. These mappings are based on community best practices, and they can be changed by editing configuration files. XML content that can be easily mapped to RDF (e.g. Dublin Core metadata and the contents of the RELS-EXT datastream) are transformed into RDF properties before being imported into Fedora 4.x. Any managed datastreams are stored as binary resources in Fedora 4.x, and external content can optionally be fetched and stored in this manner.

The primary advantage of migration-utils is its agnosticism toward front-end applications. It is designed to maintain the basic structure of Fedora 3.x data in Fedora 4.x with some XML to RDF transformations where appropriate. The application also has a number of configuration options and supports customization via plugins that could be written for specific use cases. The tool could potentially save institutions the effort of writing custom data migration scripts, particularly if they are using a custom front-end environment that would not be able to take advantage of either of the other two tools.

FedoraMigrate

FedoraMigrate was developed within the Samvera community to facilitate migrations between Fedora 3 and Fedora 4 repositories within the context of Sufia, a popular Samvera institutional repository application. FedoraMigrate iterates over an existing Fedora 3 application using the Rubydora gem. For each object it encounters, it creates a new object with the same identifier in Fedora 4 and migrates each datastream, including any versions, and verifies the checksum of each. Permissions and relationships are migrated as well. The migration process takes place in two steps: first, the resources are migrated, and then the relationships are added.

FedoraMigrate is capable of transforming XML-based metadata in Fedora 3 to RDF properties in Fedora 4; however, the mappings for each metadata element must be defined in the tool's configuration, which could be time consuming. In general, the tool is configurable, but this configuration must be done in Ruby code, so a developer with Ruby experience will need to configure and run the migration. FedoraMigrate was written with Sufia in mind, so it would need to be customized to support other Samvera applications.

Migrate_7x_claw

Islandora 8 (CLAW) makes use of the Drupal Migrate API to provide tooling for migrations from Fedora 3.x Islandora installations to Islandora 8.x. The Drupal Migrate API provides services for migrating data from different sources to Drupal 8; plug-ins can be written to support different migration use cases. The Islandora community developed the migrate_7x_claw module based on this API, which includes plug-ins for different types of data stored in Fedora 3.x-based Islandora installations.

While this module can be run from the command line using Drush, it can also be accessed via the user interface, making it easier for repository administrators to use. Configuration is relatively straight-forward: the user simply enters the base URLs for their Fedora and Solr instances, the username and password for Fedora, and a Solr query to find and retrieve the objects to be migrated. These queries can be based on content model or anything else that might be indexed in Solr.

The relative uniformity of Islandora installations will make this tool quite useful to the community. Users who have customized the defaults will need to make some configuration changes, but in general this tool should help most Islandora 7.x users easily move their data into an Islandora 8.x repository.

Usefulness of Tools for Migrating Sample Data from Institutional Profiles

Islandora

Three of the profiled repositories (Florida State, Michigan State, and Williams College) use Islandora 7.x as their repository framework. Like most Islandora institutions, none of them have made heavy customizations to their repositories, though in some cases they have added custom datastreams to the standard Islandora content models. However, these custom datastreams will simply be exported and imported by the migrate_7x_claw tool as media in Islandora 8.

The main issue with Islandora is not the migration tool, but the lack of full support for Islandora 7.x content types in Islandora 8. Specifically, paged content (e.g books and newspapers) is not yet supported (though a module has been developed by the community and should be incorporated soon) and correspondingly there is no support for paged content in the migration

tool. However, once paged content is supported in Islandora 8, it should be relatively straightforward to support this content in the migration tool.

Samvera

The FedoraMigrate tool is specifically designed to work with the Sufia Samvera application, and therefore would only be useful to institutions making use of this application (which has since been superseded by the Hyrax application). While the migration tool could certainly be updated, it has not received any substantive code commits for over two years. Even if the tool were to be updated to work with Hyrax, which is similar to Sufia, it would not be useful to institutions like Stanford that have heavily customized both their Samvera applications and their data models. A migration to any new system would likely need to be done in a customized, in-house way at Stanford.

Custom

Of the three available tools, migration-utils would be the most useful to the custom Fedora 3.x repositories (National Library of Medicine, University of Wisconsin-Madison, UNC Chapel Hill, Amherst College). While it won't address any of their front-end applications, migration-utils could be helpful in simply getting the data from Fedora 3.x to Fedora 4.x. In each case this would require some configuration and likely customization via plug-ins, but it would save the effort required to write custom migration scripts. However, the tool has not had any releases since Fedora 4.6.x so it would need to be updated to support Fedora 5.x and higher.

Gaps and Analysis

Of the currently available migration tools, migrate_7x_claw is the most robust and well-supported with greatest opportunity to impact a large number of institutions in the Fedora community. As more content types are supported, a greater number of Islandora repositories will be able to be migrated to Islandora 8. With over 260 installations around the world running on Islandora 7.x and Fedora 3.x, this represents an enormous opportunity for the Fedora and Islandora communities.

Migration-utils is a useful tool in principle, but it is hampered by a lack of updates and its support for generic migration use cases. However, this represents a potential opportunity for the Fedora community to improve the tool based on the migration needs of those with custom front-end implementations. While it wouldn't be possible to develop a tool that will work out-of-the-box in every scenario, a focus on configurable property mappings and data transformations could make the tool much more useful to the community.

Migration Tools Conclusion

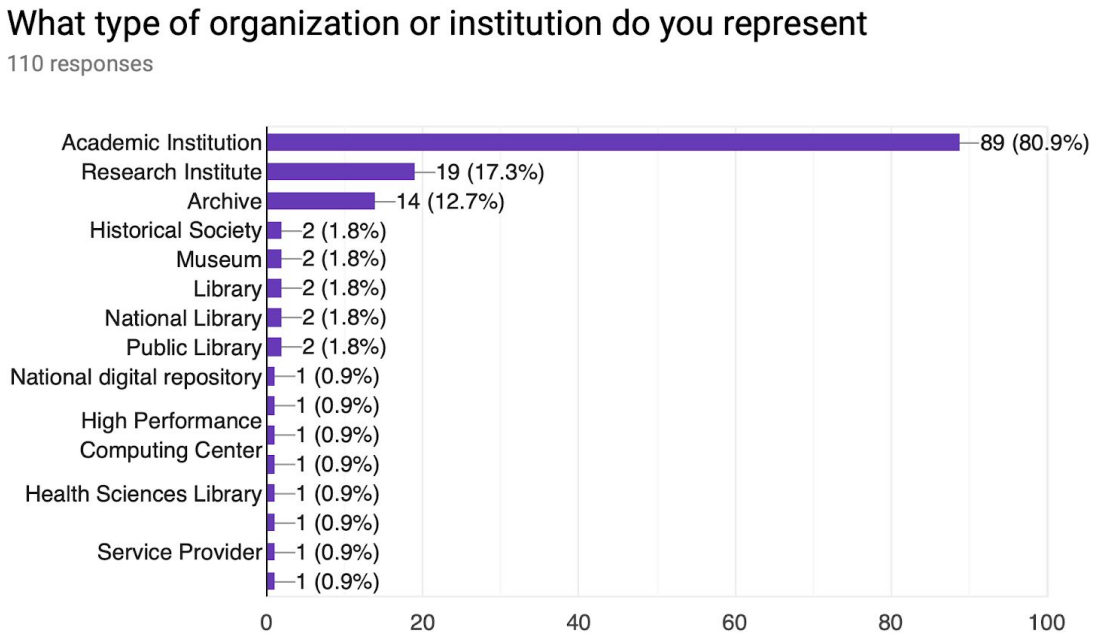
While the Islandora community has taken longer to release a version of Islandora that supports Fedora 4.x and higher, their use of Drupal and a common application framework has given them

a huge advantage in terms of developing migration tools that will support a majority of use cases in the Islandora community. The greatest gaps in support are therefore with custom Fedora 3.x repositories and those that are using Samvera tools but not a common application like Sufia or Hyrax. By taking migration-utils as a starting point and gathering requirements for improvements it would be possible to support a greater number of migration projects throughout the community.

Designing a Migration Path Survey Summary

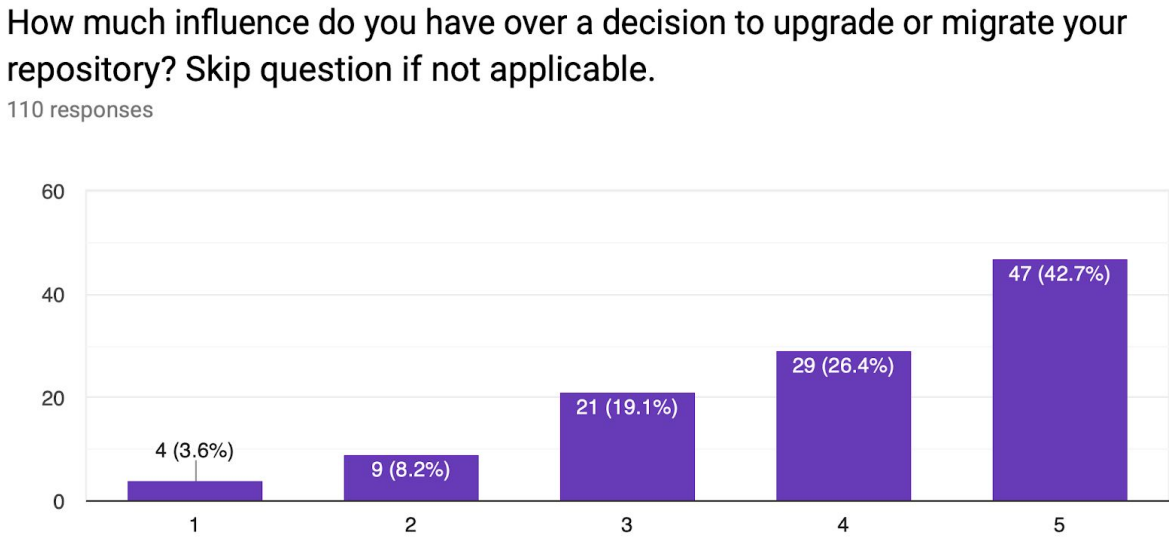
The Designing a Migration Path survey was created to assess the barriers to migrating to a supported version of Fedora. The survey closed on May 8, 2019 with 111 responses. This document summarizes the responses and provides some brief analysis.

The majority of represented institutions are Academic (80.9%) or Research Institutions (17.3%), primarily located in North America, specifically the United States (72.7%), while the remaining respondents were from other global institutions.

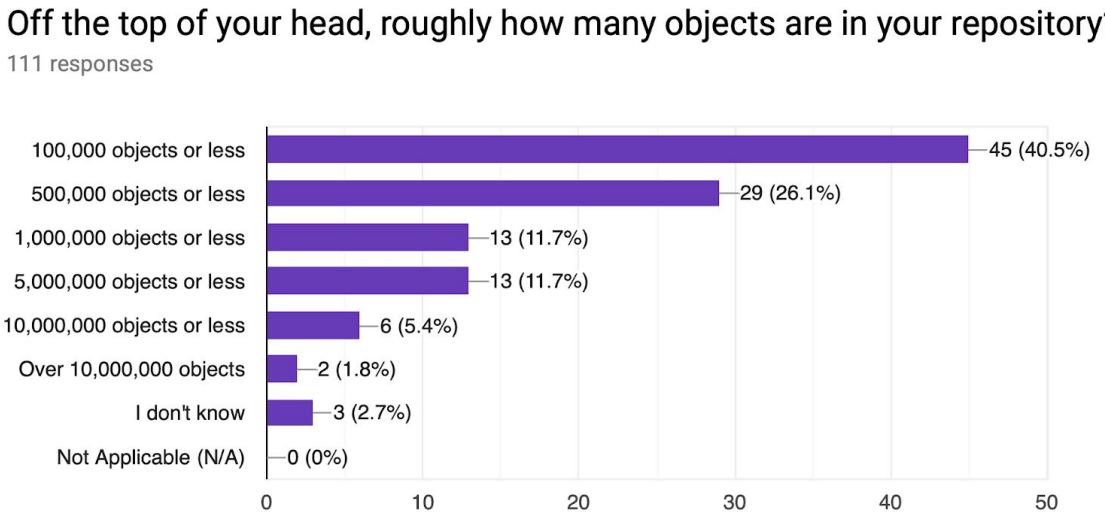


Respondents primarily identified themselves as holding one of four roles in their institutions, such as Developer/Software Engineer (40.5%), Repository Manager (37.8%), Librarian (27.9%), or System Administrator/DevOps (25.2%). Further, of the 111 respondents, 47 (42.7%) reported that they had a great deal of influence over their institution's decision to either migrate to

another application or update their repository platform.

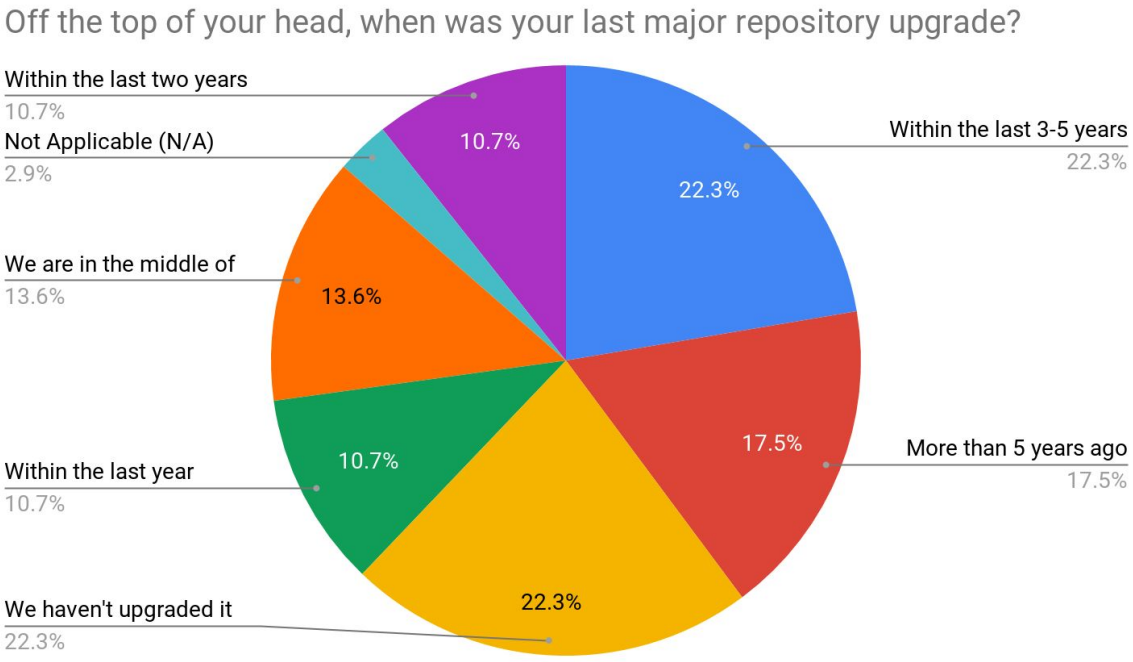


The most common forms of content stored in these repositories include cultural heritage and special collections data (85.6%), archival materials (77.5%), scholarly publications (56.8%), and research data (46.8%). Of these repositories, many (40.5%) manage 100,000 items or less, with a few (11.7%) hosting 5,000,000 items or less .

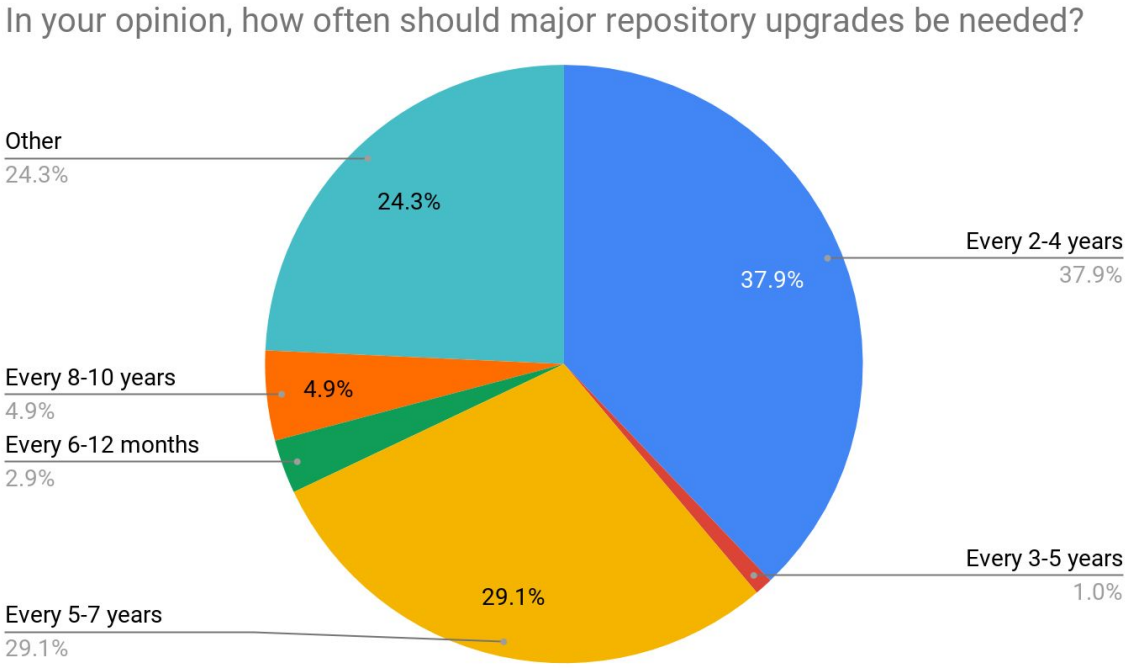


Many respondents (45%) reported having the equivalent of 1-3 full time employees working on the repository, compared to 18.9% of respondents with 4-6 FTE and 11.7% with less than one FTE. The overwhelming majority of respondents (92.8%) have a repository based on Fedora, and/or Islandora, and/or Samvera.

When asked about the most recent upgrade to the platforms that their institutions were working with, the majority of respondents reported that they either had never upgraded repository platforms, or their upgrades were 3-5 years old, accounting for 44.6% of survey respondents.



Respondents also overwhelmingly declared that upgrades should be required at most every 2-4 years (38.8%), with a large number of respondents preferring 5-7 years (29.1%). However, there were many write-in responses to this question with various versions of “it depends” or “I don’t understand what major upgrade means”.



Participants were asked to choose, in their opinion, the three issues with the greatest impact on major repository upgrades. Of the 14 offered options, the three most pressing barriers to upgrading repository platforms include:

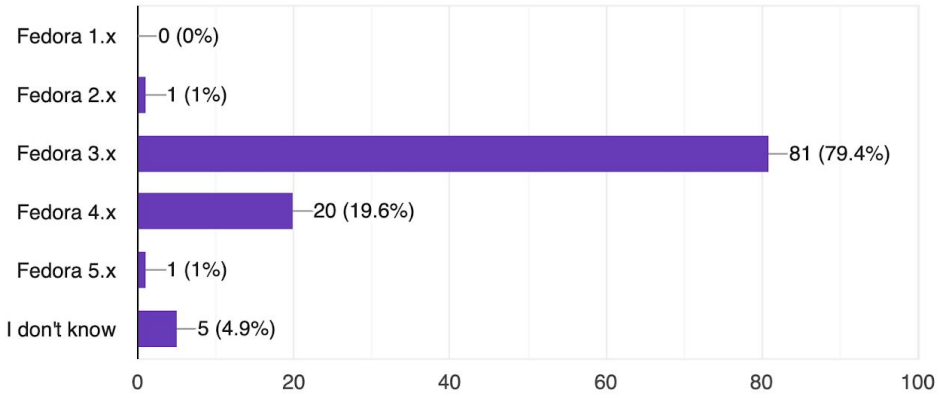
- Availability of the resources (staff or funding) (68%)
- Maturity of the software (61.2%)
- Architecture and design of the application (35%).

The results of this question identify the most significant issues the Fedora community faces when upgrading their versions of Fedora.

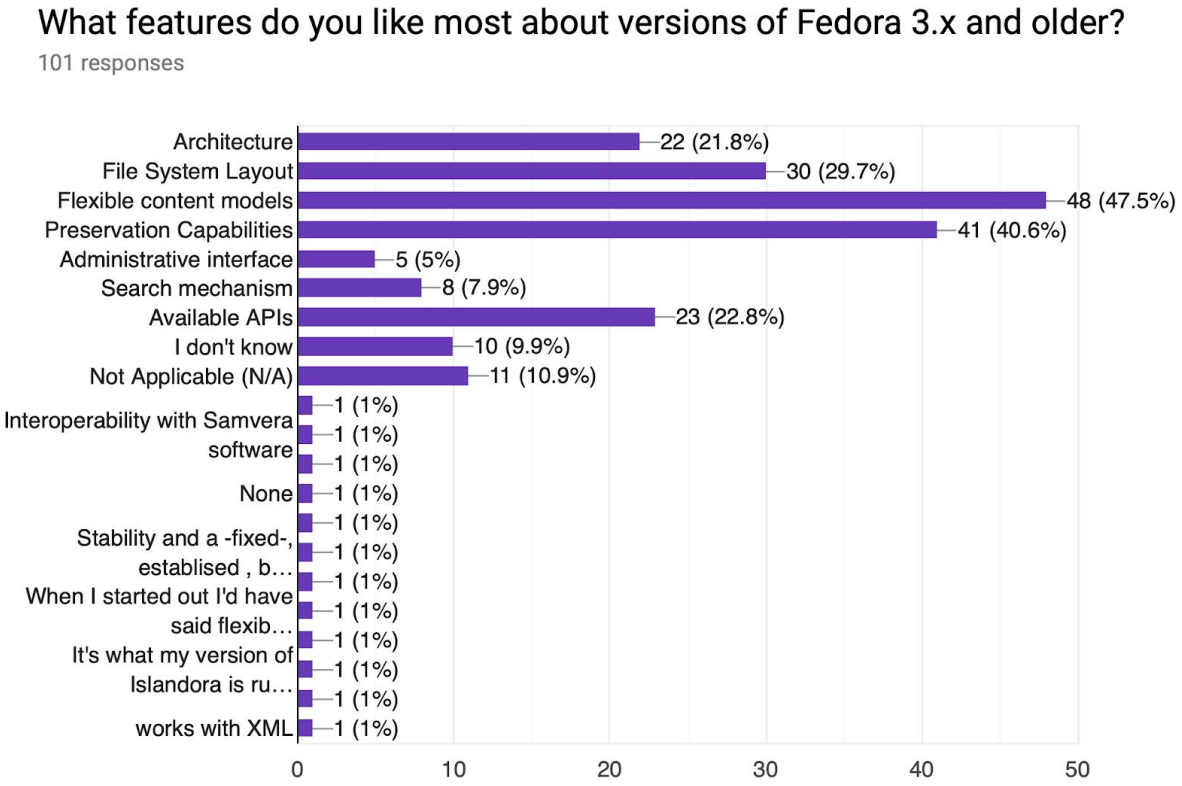
Participants were also asked which version of Fedora their institutions currently use. The majority of respondents reported still using Fedora 3.x (79.4%), with only 20.6% using Fedora 4.x or higher.

To the best of your knowledge, what version of Fedora repository software are you using currently? Skip this question if not applicable.

102 responses



Participants were again given several options and asked to pick their favourite features of Fedora 3.x and older versions. Of the 20 options for features, participants selected; Flexible content models (47.5%), Preservation Capabilities (40.6%) and File System Layout (29.7%) as the most valued features of Fedora 3.x and older.



When asked a similar question about the most valued features of Fedora 4.x, survey respondents replied that their preferred features were Linked Open Data Support (32.7%), External Component Integrations (25.5%), Focus on Existing Standards (23.5%), while 23.5% of respondents replied “I don’t know”.

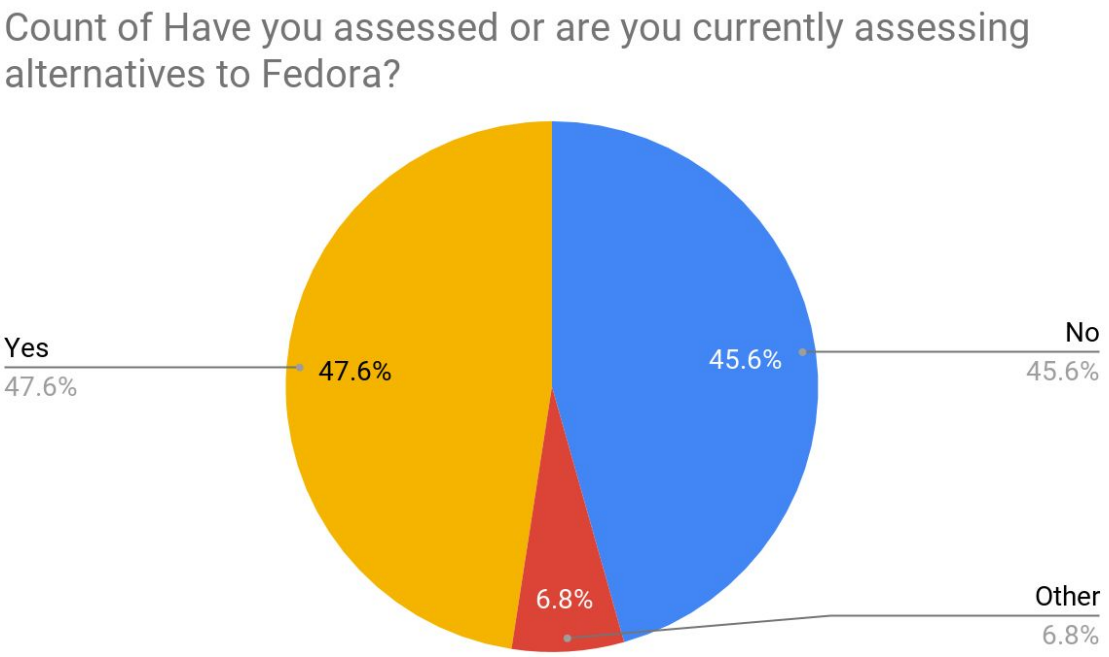
Participants were then asked explicitly about the barriers they had experienced in attempting to migrate to Fedora 4.x or higher. The most pressing barriers reported in responses to this question were;

- Insufficient Time/Staff/Resources (48.5%),
- Lack of Compatibility with Front End Applications (44.7%),
- Change in Metadata Standards for Description (39.8%) and finally
- Issues with Performance and Scale (33%).

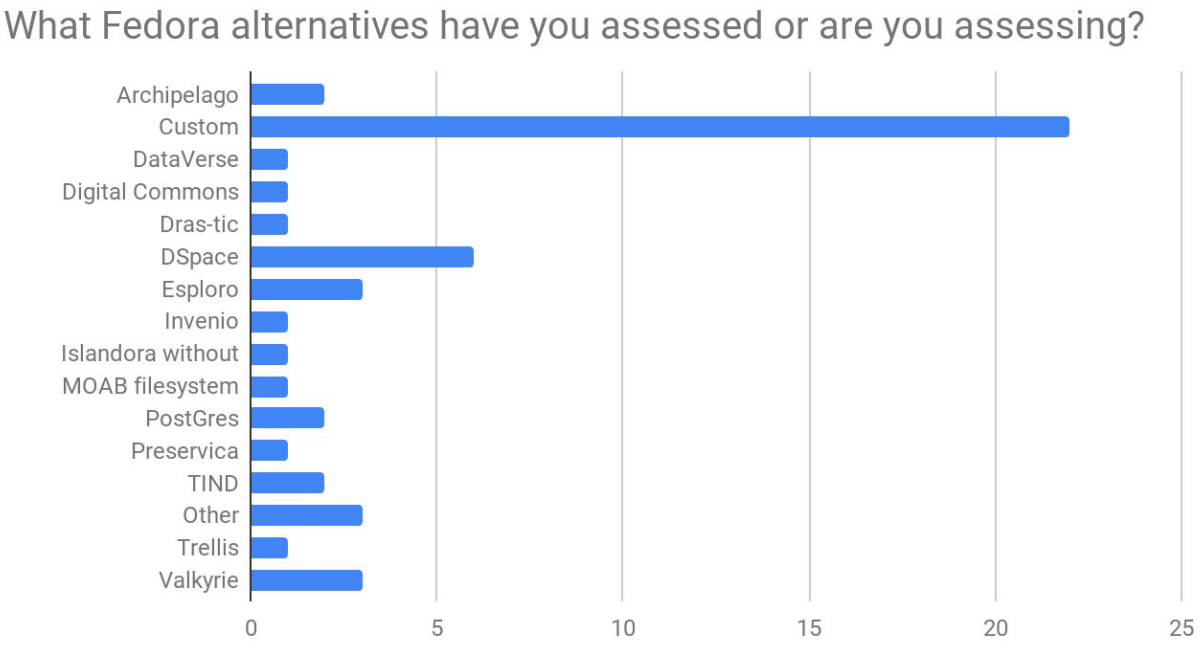
Following this question, participants were asked what would help them upgrade their version of Fedora. The top three responses are

- Content Migration Tools (70.5%),
- Metadata Migration Tools (61.4%), and
- Documentation (52.3%).

Responses to the next question, meant to determine if the respondents had considered migrating away from their Fedora platform, were split fairly evenly: 47.6% answered “Yes” while 45.6% answered “No”.



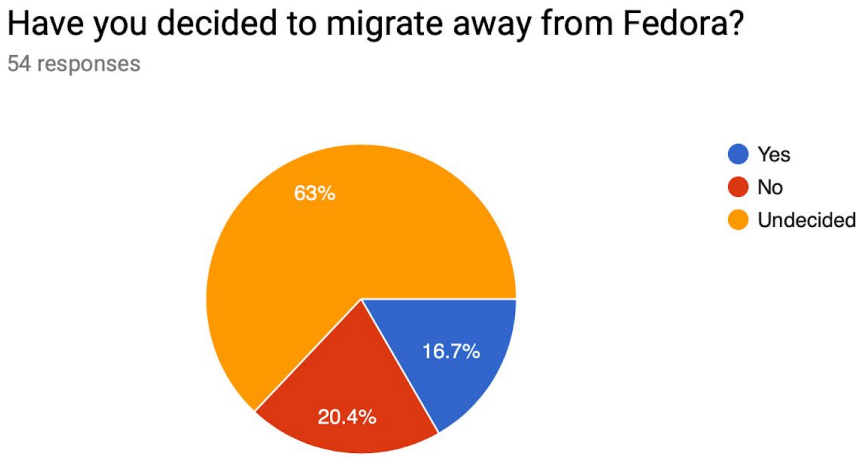
For those who have considered alternatives to Fedora, most looked into custom solutions, but many other alternatives were mentioned.



When asked why they were assessing alternatives to Fedora, respondents wrote in a variety of answers. However, three primary themes are present in the responses:

- Because it doesn't meet our needs (~55%),
- Because it's a completely different platform (~17%),
- Because of concerns over long term support (6%).

Finally, of those who are assessing alternatives to Fedora, most are currently undecided (63%) with only 16.7% of respondents having already decided to migrate away from Fedora. This represents a significant opportunity to address the concerns identified in this survey in order to keep people in the Fedora community.



Focus Group Follow-Up

63 respondents provided their contact information optionally. The grant team sorted these respondents by country and selected a group of 18 people from the USA to invite to a secondary group consultation. This consultation was scheduled for one hour and conducted as an online conference call; 11 of the invited respondents were able to attend.

In order to provide context for the discussion, participants were provided with a list of questions to be explored on the call:

1. Many respondents indicated that content and metadata migration tools would help them migrate to a supported version of Fedora.
    a. What would such tools need to do in order for them to be useful to you?

    b. On what timeline could you begin using them?
    c. What approvals would be needed?
2. What other barriers to starting the migration exist? What other migration support would you need? We're trying to understand whether or not these tools would be used if we built them, and if there are timeline sensitivities.
3. Linked open data support was reported as the most liked feature of Fedora 4+. How are you using linked data at your institution? How do you plan to work with linked data in Fedora?
4. The Fedora 3 filesystem layout was cited as a popular feature. Does the Fedora 6 design, with its focus on the Oxford Common File Layout, appeal to you? Would you adopt this new version of Fedora? What's missing from the design?

Each of these questions was discussed amongst the participants, and the results have been summarized below.

While content and metadata migration tools were identified as being potentially useful, the group highlighted other significant barriers to migration that would not be mitigated by tooling. A prime example is data models: Fedora does not prescribe or enforce data models, so implementers are looking for guidance on how to transition their existing Fedora 3.x data models to Fedora 4.x and higher. Documentation, case studies, and best practices would help here.

More generally, several participants agreed that if any labor would be involved in a migration it creates a decision point: should I move to the new version, remain on the current version, or choose something else? There needs to be a significant increase in value in order to justify the work involved in a migration, particularly for complex systems and/or custom interfaces. Some of the important features of Fedora 3.x are now available in a variety of applications and services (including cloud providers like Amazon AWS), so there are more options and potentially less complexity, particularly in the case of hosted solutions.

This was a nice segue into the topic of Fedora 6 and the OCFL, which provides some additional features around digital preservation. The reaction to the OCFL was generally positive, though some participants were concerned that it is not yet proven to work in a performant and scalable way in a variety of storage architectures. One participant noted that their system was quite complex so they would prefer a more performant database replacement for ModeShape rather than a return to the filesystem focus of Fedora 3.x.

While linked open data support was reported as the most popular feature of Fedora 4.x+, participants were unsure to what degree Fedora was actually being used to support linked open data use cases. It is a broad topic, and survey respondents could have had different things in mind when they selected it as the most liked feature, so the focus group participants didn't want to read too much into this result.

The issue of timelines was raised, and several participants indicated that their institutions moved slowly; they are unlikely to migrate off their current (working) system(s) until a critical issue forces them to move. There was interest in prolonging the life of Fedora 3.x in the meantime, perhaps via a Fedora 4.x+ API on top of Fedora 3.x that would allow for a more incremental migration, along with continued security updates for Fedora 3.x.

Recommendations

Based on all the components of the project, including the outcomes of the survey and focus group, the grant team drafted a set of recommendations that reflect two important themes that arose over the course of the project: effort and value. Simply put, any software upgrade (including a migration) that requires significant effort, either in the form of local human labor or financial costs, creates a decision point. The institution needs to decide whether to go forward with the upgrade, stay on the current version, or move to another platform, and the answer all depends on value. The new version of the software must provide enough additional value compared to both the current version and other available software packages in order to justify the cost of the upgrade. In the case of Fedora, many survey and focus group respondents indicated that they did not perceive enough additional value in Fedora 4 or 5 to justify the relatively high level of effort required to upgrade from Fedora 3. Therefore, our recommendations focus on increasing the value of Fedora while decreasing the level of effort required to upgrade:

1. Develop and deliver Fedora 6 with support for transparent persistence (preservation).
2. Develop migration tools to support upgrades from Fedora 3, 4, and 5 to Fedora 6.
3. Test Fedora 6 at scale under a variety of deployment scenarios, including cloud-based deployments.
4. Work with pilot institutions to document case studies, data model examples, and best practices for each community moving to Fedora 6.
5. Communicate the need to upgrade from Fedora 3.x to avoid stability and security issues.
6. Communicate the value of migrating to Fedora 6 in several contexts:
    a. As an open source, community supported project.
    b. As insurance against vendor/hardware lock-in.
    c. As part of a portfolio of open source programs supported by LYRASIS.
7. Pilot a repository upgrade service through LYRASIS.

The first recommendation is based on value; Fedora 4 and 5 focused largely on implementing a standards-based API with native support for linked data. This focus took attention away from Fedora's preservation features, particularly transparent file persistence. By introducing a more standardized version of this feature (using the OCFL) in Fedora 6, we will increase the value of Fedora for digital preservation use cases. This file-based persistence will also provide a new

migration path that will lower the amount of effort required to migrate to a supported version of Fedora: it will be possible to transform Fedora 3 content on-disk into Fedora 6-compatible content without exporting and importing everything through the API. The release of Fedora 6 must therefore be coupled with the appropriate tooling to ensure that the community can move forward with reduced effort.

At the same time, Fedora 6 must be tested at scale under a variety of different deployment scenarios, including cloud-based deployments. As institutions, particularly in the United States, increasingly adopt a cloud-first policy, Fedora must be able to keep up with these trends and perform well regardless of how it is deployed. In particular, Fedora 6 will provide additional value in a proprietary cloud-based environment by persisting content in an application and storage agnostic way, employing open standards and open data, thereby ensuring continued access to the raw data and mitigating against vendor lock-in. We look at the implementation of Fedora 6 in the cloud as a form of insurance against increasing dependencies on proprietary systems.

The survey and focus group results also made clear that libraries and other similar institutions move slowly and cautiously, so it is important to document case studies, data model examples, and best practices that can serve as a roadmap for institutions considering a move to Fedora 6. The project team recommends working with pilot institutions from representative communities (e.g. Islandora, Samvera, custom) to validate the design and development of Fedora 6, document their migration to the new platform as case studies, and provide community guidance in the form of best practices learned during the pilot process.

Communication is another important recommendation related to value; as a community supported open source project Fedora provides a number of benefits, including a global network of users and contributors, transparent code and processes, and insurance against vendor and hardware lock-in. This is particularly true of Fedora 6, which will include application-independent persistence via the OCFL. As one of a portfolio of open source programs supported by LYRASIS, Fedora benefits from the stability and commitment of an organizational home with a long history and proven track record in the memory institution community. Under the LYRASIS umbrella, future Fedora development can take advantage of opportunities to align with the goals of other LYRASIS programs to pursue a coherent long-term strategy for managing digital resources.

Finally, the project team recommends piloting a repository upgrade service for a variety of platforms that could be offered by service providers. This would help lower the barrier to major repository upgrades by providing access to a team of experts that could supplement or take on a repository upgrade project for institutions that lack sufficient local resources. This service could be seeded with grant funding initially to help with startup costs (hiring staff, preparing resources, building tools, running pilots, etc.), with a long term plan to pay for itself through service revenue.

References (for Environmental Scan)

1. Bailey, Greg et al. Who gives a DAM?: The Iterative Process for Assessing Digital Asset Management Tools.
2. Berghaus, Frank et al. CERN Services for Long Term Data Preservation.
3. Bridge2Hyku Project Team. Digital Collections Survey Report.
4. Bullen, Andrew. A Doomsday Scenario: Exporting CONTENTdm Records to XTF.
5. Conrad, Suzanna. Spinning Communication to Get People Excited about Technological Change.
6. Fallaw, Colleen et al. Overly Honest Data Repository Development.
7. Fedora Leadership Group. Fedora and Digital Preservation Survey.
8. Gilbert, Heather and Mobley, Tyler. Breaking Up With CONTENTdm: Why and How One Institution Took the Leap to Open Source.
9. Hardesty, Juliet L. and Young, Jennifer B. The Semantics of Metadata: Avalon Media System and the Move to RDF.
10. Mayo, Dave and Bowers, Kate. The Devil's Shoehorn: A case study of EAD to ArchivesSpace migration at a large university. The Devil's Shoehorn: A case study of EAD to ArchivesSpace migration at a large university.
11. Neatrour, Anna et al. A Clean Sweep: The Tools and Processes of a Successful Metadata Migration.
12. Nowak, Marcin et al. Objectivity Data Migration.
13. OCLC Research. International Linked Data Survey for Implementers, 2018 Report.
14. Rochkind, Jonathan. On the present and future of samvera technical architectures.
15. Sacchi, Simone and Cunningham, Eva T. Migrating an IR to New Technology: Opportunities, Challenges, and Decision-Making Processes.
16. Simic, Julia and Seymore, Sarah. From Silos to Opaquenamespace: Oregon Digital's Migration to Linked Open Data in Hydra.
17. Soper, Devin and Brown, Bryan. Migrating to an Open Source Institutional Repository: Challenges and Lessons Learned.
18. Stein, Ayla and Thompson, Santi. Taking Control: Identifying Motivations for Migrating Library Digital Asset Management Systems.
19. Stein, Ayla and Thompson, Santi. Understanding Metadata Needs When Migrating DAMS.
20. Tripp, Erin. Open Source Repository Upgrades: Top Advice from Practitioners.
21. Tripp, Erin. Reframing Open Source Repository Upgrades.
22. Van Tuyl, Steve et al. Are we still working on this? A meta-retrospective of a digital repository migration in the form of a classic Greek Tragedy (in extreme violation of Aristotelian Unity of Time).
23. Weidner, Andrew et al. Outside The Box: Building a Digital Asset Management Ecosystem for Preservation and Access.
24. Wiseman, Christine and Matthews, Al. Time, Money, and Effort: A Practical Approach to Digital Content Management.

25. Witkowski, Alan et al. Massive Newspaper Migration — Moving 22 Million Records from CONTENTdm to Solphal.
26. Wu, Annie et al. Hitting the Road towards a Greater Digital Destination: Evaluating and Testing DAMS at the University of Houston Libraries.
27. Yeh, Shea-Tinn et al. Deploying Islandora as a Digital Repository Platform: a Multifaceted Experience at the University of Denver Libraries.