Holistically Preserving and Presenting Complex Research Data

Chad M. Mills

Rutgers University

cmmills@rulmail.rutgers.edu

Abstract

Digital repositories are being continually challenged by the complexities of research projects and their associated resources. In the research data environment the role of the digital repository is multi-faceted, requiring storage, preservation, security, and high-availability of resources. Researchers expect complete research data project access to a wide array of data files and associated documentation.

This paper discusses the approach taken by the RUcore, Rutgers University Community Repository by holistically storing and managing individual resources that have complex file structures.

Introduction

RUcore began accepting research projects and their digital resources in 2010. Initially the resources were simple in nature, typically comprising a single file for each data resource. In such a scenario, applying time-tested data models and storage rules borrowed from document and image resources worked well. However, as time passed and RUcore was exposed to a multitude of multi-disciplinary research projects, the RUcore team realized that resources derived from research projects can be highly complex, requiring their own data model and storage rules.

What is a Complex Resource?

For the purposes of this paper we will establish what makes a resource complex. Pragmatically speaking, a resource is complex when a resource and its associated files need to be archived and presented in a structured hierarchical manner. From a functional point of view, when a researcher provides a directory of files and sub-directories, the researcher and other users of those objects expect to be able to download that resources files in exactly the same format, referencing the same filenames, and following the same file or directory structure that it had before it was accepted into the repository.

These complex resources can manifest themselves in a variety of ways. They can be a collection of sensor data and calibration settings from an instrument, or proprietary software developed for a research project. A complex resource could also be a series of data files used for statistical analysis, spreadsheets, and/or documents. Throughout the life-cycle of a research project multiple versions of these complex resources will exist. All of these resources need to be related to the research project, and in some cases, to each other.

Possible Strategies for Handling Complex Resources

We investigated how other research/institutional repositories were handling rich complex resources. Generally, we found two approaches are being implemented.

The first approach stores all files associated with a complex resource as a single archive file. ZIP and/or TAR formats are the most popular formats. The archive file is then part of a single repository object, which forms the complex resource. This strategy preserves the original file structure well, but offers little flexibility when accessing the files. Accessing individual files of the resource is not an option; the user can only download the entire resource.

The second approach stores all files individually, as discrete repository objects, and then offer a mechanism to bind the objects together to form the original complex resource. Resource Descriptive Framework (RDF) statements are used to store the objects' file structure. This is a very granular approach and, depending on the size and complexity of the resource, can require a good deal of computational effort when reassembling the resource. This approach can also be very demanding of the researcher's time and energy, usually requiring the researcher to provide some type of file-level metadata.

Both approaches have drawbacks that are unappealing and not ideal for our research community. We need to offer a balanced solution that would place as little burden as possible on the researcher, while still offering advanced access to the complex resource. Without this, researchers would go elsewhere -- or in some cases nowhere -- with their research data.

Using the METS Structural Map to Capture Complex File Structures

The structural map is the heart of a METS document. It outlines a hierarchical structure for the digital library object, and links the elements of that structure to content files and metadata that pertain to each element.[1]

OpenWMS is an open source workflow management system we developed and use to generate METS-compliant metadata, create presentation derivatives and ingest resources in RUcore. We have been selectively using the structural map portion of the METS schema, mostly to describe the table of contents for books.

The METS structural map is ideally suited for preserving the file structures and directories of complex research resources. Capturing a resource's complex file structure in a METS structural map and storing it as part of the resource provides a map that can be used when binding the resource back together. By using the METS structural map as our mechanism to render a complex resource's files in their original format, we can store the resource's files in any manner we wish. See Attachment 1.

Applying this method of storing complex file structures, we are able to store the resource files in a simple manner, allowing us to store the resource as a single repository object. This reduces any pressure on the researcher to provide file-level metadata. The researcher or metadata creator simply describes the complete resource, not every file that is part of the resource.

When the stored complex resource is requested, we render the resource on-the-fly using the structural map to determine the file structure and present the user with a single archive file. Creating

these archive files on-the-fly also allows us to offer the researcher/user an option to preview which files they are downloading. The researcher/user has the opportunity to deselect files or folders they do not wish to download. See Attachment 2.

Using RDF Technology to Associate Complex Resources

While we are not using RDF to store and manage complex resources as discrete repository objects, we are using RDF statements to associate complex resources within a research project, and to express resource relationships.

Every research project has a corresponding resource stored in the repository that describes at a high-level the background of the project, owners, stakeholders, etc. That project resource serves as the nucleus for all complex resources associated with the project. Using RDF statements the complex resources are associated with the project resource.

This network of resources allows the researcher/user to download the entire project, much as he or she would download a single complex resource. Using a combination of the RDF statements and structural maps, the researcher/user has the ability to select for download one or more complex resources, or selected files from a complex resource.[2]

The final delivered archive file is portable, structured and highly reusable. For quality assurance and data integrity purposes, file checksums are delivered as part of the archive file. The archive file follows the BagIt hierarchical packaging format.[3]

Conclusion

The objective driving the development of this approach to storing and delivering research data is that the researcher need not feel overwhelmed by the process of delivering research projects and complex resources for inclusion in the repository. We have found that by implementing a series of standard solutions and intuitive interfaces we can find a balance that makes the process more palatable to the researcher.

A technical description outlining how the solution was implemented is available at the following location.

`http://rucore.libraries.rutgers.edu/collab/ref/spc_sawg_r7_0_file_hierarchy.pdf`

References & Examples

[1] METS Structural Map [website] 3/2/2013
http://www.loc.gov/standards/mets/METSOverview.v2.html#structmap

[2] Project Download [website] 3/4/2013
http://rucore.libraries.rutgers.edu/research/results.php?rtype[]=collection&q1=Primate

[3] BagIt Packaging Standard [website] 3/2/2013 http://en.wikipedia.org/wiki/BagIt

Attachment 1

Sample Structural Map

```
<?xml version="1.0" encoding="utf-8"?>
<METS:mets xmlns:METS="http://www.loc.gov/METS/">
        <METS:structMap ID="SMAP-PRES" TYPE="logical" LABEL="File structure">
                <METS:div ID="div1" LABEL="zip, pdf and other files" ORDER="1" TYPE="folder">
                        <METS:fptr ID="FILE0001" FILEID="{resourceID}" CONTENTIDS="ZIP-1"/>
                        <METS:fptr ID="FILE0002" FILEID="{resourceID}" CONTENTIDS="ZIP-2"/>
                        <METS:div ID="div1.1" LABEL="pdf" ORDER="2" TYPE="folder">
                                <METS:fptr ID="FILE0003" FILEID="{resourceID}" CONTENTIDS="PDF-1"/>
                                <METS:fptr ID="FILE0004" FILEID="{resourceID}" CONTENTIDS="PDF-2"/>
                        </METS:div>
                        <METS:fptr ID="FILE0007" FILEID="{resourceID}" CONTENTIDS="TIF-1"/>
                        <METS:fptr ID="FILE0008" FILEID="{resourceID}" CONTENTIDS="DBF-1"/>
                </METS:div>
        </METS:structMap>
</METS:mets>
```

Attachment 2

User Interface to Select Individual Files from a Complex Resource