Orbital Project: Technical Specification

Nick Jackson
nijackson@lincoln.ac.uk

Abstract

Orbital is a research data management project at the University of Lincoln. This technical specification outlines the initial user requirements, rationale for design decisions, and provides a specification for the initial feature set of the project's storage and management systems.

Contents

1 User Analysis

Analysis is based on our initial case studies[1], individual user and stakeholder interviews, group meetings and observation of research. The process of user analysis is ongoing throughout the project.

1.1 Existing Data Formats

Existing research data exists in a number of data formats both structured and unstructured. Creating an exhaustive list is impossible given the scope for variations and nuances of different projects, but a sample of data formats includes:

- Excel spreadsheet files
- CSV files
- XML files
- Proprietary binary format
- Plain text unstructured files
- Plain text structured data files
- Audio waveforms as .wav files
- Images as .png and .jpg files
- Research-relevant data embedded in file names
- Research-relevant data embedded in folder structure
- Application source code (in various languages)
- Free-form text reports as various formats, including:
  - PDF files
  - Microsoft Word .doc and .docx files from various versions
  - HTML accessible via a web interface

It is worth noting that although some of these files contain embedded metadata regarding their contents, in some instances metadata is contained in external files or embedded in folder structure or file names. In many cases files will contain metadata regarding the file itself (such as creation and modification time), and although this metadata is not necessarily relevant to the data being stored it should be recorded.

1.1.1 Format Translation

It will be necessary to develop 'translations' between these various discrete file formats (where they express structured data) and the Orbital data storage format. These 'translators' may be generic in nature (such as those which can transform Excel spreadsheets and CSV files) or specific to individual projects (such as those handling proprietary binary formats).

It may also be necessary, specifically where current research relies on a specific file format, to implement 'translators' in the opposite direction, i.e. converting from Orbital's own data storage format back to a defined discrete file. As above these may be generic in nature, or crafted on a per-project basis. Further exploration of the technology for interpreting the contents of discrete files is given in section 3.4.2 on page 7.

1.2 Existing Data Storage

At present the storage of research data (based on initial case studies[1], user interviews and observation of research) is managed by the researchers and their research partners using a variety of means with varying degrees of organisation, accessibility, management, integrity, versioning and resiliency. This ranges from a simple storage of files in a flat file structure, through using file names to include version data or nesting folders based on data type through to online document repositories, the use of shared database technologies or web-accessible bespoke data portals. Occasionally these methods are supplemented using other technologies, such

as the use of Dropbox[1] to provide a synchronised workspace across different researchers, or relying on the University's network drive provision to perform backups.

Within the scope of engineering the size of research datasets ranged significantly from a few tens of megabytes through to hundreds of gigabytes. In the case of one research project the 'working' dataset, although only a few gigabytes in size, was taken from a larger dataset running into terabytes of data.

There is an understanding amongst most academics interviewed that backups of research data are required, although generally this is performed through manual copying of data to a second device such as a laptop or memory stick, as opposed to through the use of a dedicated backup solution or storage platform with a defined backup policy.

It is worth noting that none of the academics interviewed or observed made use of a natively versioned filesystem or storage platform, instead opting to manually manage discrete versions of their work through saving under an alternate file name or copying a data source directory.

1.3 Current Repository Solution

The University of Lincoln Library currently provide and support an institutional repository[2] using the ePrints[3] platform. This platform allows for the curation and long-term storage of research data objects in discrete file format or by reference to an external data source. Its primary purpose is to store the research output from the University – for example research papers – a task which is outside the scope of Orbital.

The current repository is curated by members of the Library staff, and involves a manual approval process. Any integration with Orbital must honour this manual approval process for inclusion in the curated catalogue and any subsequent changes including versioning of files.

The Library requires certain metadata to be present in order for a work to be included in the Repository. The required metadata broadly overlaps with the metadata required by the OAIS Reference Model to enable robust archiving and retrieval of data[2], and those suggested by the DCC Digital Curation Manual[3].

1.4 Research & Bid Management

The University of Lincoln's Research and Enterprise Services department are currently in the process of procuring and commissioning a platform for the management of research funding calls, bids and resource allocations. It is intended that this platform will expose APIs which allow Orbital to interact with it where necessary or desirable, although without this platform in place it is impossible to fully evaluate any implementations or identify shortcomings which Orbital may be able to fill.

2 System Requirements

It must be noted that the Orbital project is being designed and built using Agile development methodologies which rely on a constant loop of user feedback and requirements gathering[4]. This section is not a complete or exhaustive list of user requirements, and these requirements will be subject to change or extension over the lifetime of the project.

2.1 Functional Requirements

At a high level, the Orbital project is required to be able to store and manage research data generated for and as part of research projects at the University of Lincoln (and by extension at other institutions which have similar research projects). Throughout the project development and pilot the primary users will be researchers at the University of Lincoln's School of Engineering[4], although where suitable for immediate storage and dissemination the project may also include data from other schools and departments.

[1] http://www.dropbox.com/
[2] http://eprints.lincoln.ac.uk/
[3] http://www.eprints.org/
[4] http://www.lincoln.ac.uk/engineering/

2.1.1 Projects

Orbital is required to organise research data into discrete bundles categorised as "projects". Each project will represent work towards a discrete research output or collection of outputs, although these outputs may not be fully defined at the point of a project being created.

A project must contain high-level metadata, including information on the project's data management plan (DMP). Project metadata must 'trickle down' to any data which is stored under that project.

It should be possible for storage elements to exist independently of a project, which can then be moved to a project at a later point should they become relevant.

2.1.2 Filesystem Storage ('Workspace')

Orbital must provide a storage location for researchers which can be treated either as part of or as an extension of the local file system. This workspace may exist entirely within the Orbital infrastructure and require network access, or may be synchronised to the local filesystem at available opportunities. Based on requirements for speed and offline access it is proposed that the synchronisation route is used.

There is a requirement for a workspace's capacity to be sufficient to hold a project's data, which may be in excess of several hundred gigabytes as discussed in section 1.2 on the previous page. This workspace must be capable of resiliency in the event of server failure, although this is dependent on specific implementation. There is no requirement for a workspace to offer 'time-travel' functionality of the data, although this functionality could be included.

Researchers require that a project workspace both includes a personal workspace which is inaccessible to others within the project, and a shared space which can be accessed by all researchers involved in the project.

2.1.3 Discrete File Storage ('Archives')

Orbital must be able to store discrete files independently of a workspace, allowing them to be given appropriate metadata enabling their curation, access and long-term preservation.

Data must be able to be moved from a workspace into archive storage, and vice-versa.

There is a requirement for archive storage to be able to hold files of a significant size, although where possible large datasets should exist in a working dataset snapshot. These files may be in excess of several hundred gigabytes as discussed in section 1.2 on the preceding page. Archive storage must be capable of resiliency in the event of server failure, although this is dependent on specific implementation. Archive storage should be capable of storing multiple versions of discrete files and maintaining a relationship between them.

2.1.4 Dataset Storage ('Working Dataset')

Researchers require that Orbital be able to provide database-like functionality for any given set of data, which can be queried and manipulated both programmatically and via a graphical user interface.

Data must be able to be moved from discrete files (in a workspace or archive) into a working dataset, and vice-versa. Further exploration of the technology for interpreting the contents of discrete files and moving them into a working dataset is given in section 3.4.2 on page 7.

A working dataset must be able to have a 'snapshot' taken at a point in time. The data within a snapshot must be unmodifiable, although select metadata may be. A snapshot may be used as the basis of another dataset ('copying'), or representing a distinct point in time for a working dataset ('versioning'). A snapshot must be automatically made during a copy or publish operation and used as the basis of that copy or published dataset.

2.1.5 Publication

Library staff and researchers require that Orbital be able to 'publish' a file or dataset, either from archive storage or a working dataset, and that the published file or dataset has sufficient metadata to be discoverable and curatable in the context of the institutional repository.

Once published, the basic metadata for a publication must not be removable (i.e. the fact that a publication existed, along with any permanent identifiers, must persist), although the data itself should be deletable if necessary.

2.2 Workflow

The workflow of Orbital is required to be flexible enough to adapt to multiple research methodologies, as well as the preferences of different researchers or teams. For this reason Orbital should not adopt a rigid workflow pattern, and should allow researchers to build their own flows on top of the provision.

The only distinct state in a workflow which was consistently required across multiple researchers was that of "published", i.e. a point at which a discrete, fixed version of a file or dataset (or the fact that a file or dataset exists) should be made available to the general public for the purposes of citation and subsequent reuse.

3 Evaluation of Technologies

3.1 User Access Control

Given that Orbital must potentially be accessible from multiple organisations, each with their own access control systems, and that it is impossible to guarantee that an organisation will use the same authentication platform as another, user access control will be conducted using a plugin architecture which allows multiple providers of different types to be used.

Note that Orbital itself will not attempt to validate the identity or authorisation of a user to use Orbital independently of the plugin, and therefore all plugins must be trusted by the administrators to correctly identify, authenticate and authorise a user.

The exact access restrictions which are required on a per-user basis will be detailed at a later point in the project as a result of user engagement, although will initially include obvious permissions such as read, write and administrate at various levels of detail.

3.1.1 OAuth

To allow access to Orbital's APIs, Orbital will employ the OAuth 2 specification[5]. This provides a means for secured access to all API functions which can easily be implemented by third party developers and researchers wishing to manipulate their data directly, rather than using the tools provided in Orbital Manager. The latest draft of the OAuth 2 specification is currently awaiting formal ratification by the IETF, although is stable enough for general use (as evidenced by its adoption by major companies such as Google, Microsoft and Facebook).

3.2 Database Technologies

3.2.1 SQL vs NoSQL

In the design of Orbital we considered both traditional SQL relational databases (Such as MySQL, PostgreSQL and MS-SQL) and NoSQL[6] databases such as CouchDB, MongoDB, Cassandra and Hadoop. Databases were considered both in the context of storing research data itself (the "working dataset" discussed in section 2.1.4), storing metadata, and storing application operational data (such as audit logs, configuration, user accounts and project information).

The primary driving factors in Orbital's development will be the ability of the database to reliably store data of disparate types[5], to scale that data storage, and to allow rapid access to that data. Traditional SQL databases, especially those with full ACID[7] compliance, offer a high level of reliability through features such as transactions, referential integrity and write-ahead logging. Conversely, NoSQL offers improved scalability and performance at the cost of referential integrity and data durability.

[5] http://oauth.net/2/
[6] http://en.wikipedia.org/wiki/NoSQL
[7] http://en.wikipedia.org/wiki/ACID

3.2.2 NoSQL Variants

Within the NoSQL sphere there are several distinct types of database solution, offering a wide range of features and storage patterns[6]. Orbital's requirements from a NoSQL system are summarised thus:

- Ability to scale horizontally to improve storage.
- Ability to scale horizontally to improve throughput.
- Storage of data must eventually be to disk.
- Database must reach eventual consistency[8].
- Ability to store raw files.
- Ability to store documents[9].
- Ability to organise documents into discrete groupings.
- Ability to replicate data to improve resilience against failure.

Based on these requirements, and our existing experience with MongoDB in previous projects[10][11] Orbital will use MongoDB as its principal database platform for storing application operational data, metadata and the research data itself.

3.3 Storage Formats

Orbital is tasked with storing three distinct types of data. The proposed storage methods for these three types are thus:

3.3.1 Workspaces

A workspace is essentially a folder within a filesystem, for this reason it is proposed that it is stored as such. This enables easy interaction with the filesystem by the Orbital Core application if necessary (for example moving files to or from archive or working dataset storage).

It is worth noting that deployments with multiple Orbital Core servers will require that the workspace filesystem is stored on a single centrally available network drive or other shared filesystem. In all cases this shared volume should be mountable as a native volume.

3.3.2 Archives

The storage of files in the archive will be managed by a plugin system similar to that used in authentication, enabling the actual storage of files to use whatever storage infrastructure is available at the point of deployment, for example the use of cloud storage solutions (Rackspace Cloud Files or Amazon S3), RAID[12] implementations, proprietary 'big storage' solutions or other systems such as tape jukeboxes. Most implementations of archive storage will be nearline[13], and the plugin architecture will allow for this asynchronous loading of data.

In the case of solutions such as cloud storage, it may be that publicly available files can be made accessible over a CDN[14], allowing for improved access speeds. This availability will be managed by the storage plugin.

[8] http://en.wikipedia.org/wiki/Eventual_consistency
[9] Documents in this case refers to structured data storage, not an individual file.
[10] http://blog.totalrecal.org/
[11] http://jerome.blogs.lincoln.ac.uk/
[12] http://en.wikipedia.org/wiki/RAID
[13] http://en.wikipedia.org/wiki/Nearline_storage
[14] Content Delivery Network

3.3.3 Working Datasets

All working datasets will be stored exclusively in the MongoDB database. This allows for rapid access and retrieval of data, as well as complex querying of the data using a variety of criteria. MongoDB, as a so-called 'schemaless' NoSQL database is capable of storing research data in whichever format or structure makes the most sense for an individual research project and subsequently retrieving data in the same format and structure. In addition, MongoDB is capable of rapid scaling in both terms of capacity and performance through the use of sharding[15], as well as maintaining data resiliency and system availability in the event of failure through the use of replication.

3.4 Exchange Standards

3.4.1 Synchronisation of Workspaces

The protocol and standards used for the synchronisation of workspaces (as per section 2.1.2 on page 4) between Orbital and a researcher's local filesystem should include, as a minimum, the following features:

- Protocol must be open, i.e. not relying on a specific Orbital application (although one may be provided).
- Secure transmission of contents.
- Support for folders and nested folder structure.
- Partial 'delta' updates of workspace contents to reduce network traffic.
- Compression of updates.
- Tear prevention to protect against file corruption in the event of disconnection.
- Change merging at a folder level, allowing changes made in multiple locations to be reconciled.

These requirements prevent the use of protocols such as FTP and its secure variant SFTP due to lack of delta updates, compression and tear prevention, although they do support nested folders and change merging providing a suitable client is provided.

It is proposed that a version control system such as Git[16] is used to provide synchronisation between Orbital and remote workspaces. In general these version control systems provide all the required functionality, with the additional benefits of versioning support allowing researchers to roll their workspaces (or individual files) back to an earlier point in time should they wish.

3.4.2 Input of Data to Working Dataset

Orbital shall be agnostic with regards to the original input format of data to a working dataset, making use of interpreters to translate disparate data types into a standardised form which can be understood and stored by the MongoDB database layer following validation by the Core APIs. This standardised form shall be in line with the JSON data structure used in storage, and shall be further documented as part of the Orbital Core API documentation as further user requirements and use cases become apparent.

Where data is being inputted directly to a working dataset using Orbital's APIs, the data shall be in the required format as per the Orbital API documentation. Data which does not match an accepted storage format will be rejected.

3.4.3 Output of Data from Working Dataset

Similarly to the input of data, output from a working dataset can be made either natively over the APIs (in which case a representation of the stored data structures will be returned as per the Orbital API documentation), or through interpreters which will translate the stored data to a specific format.

[15] http://en.wikipedia.org/wiki/Shard_(database_architecture)
[16] http://git-scm.com/

3.4.4 API Transport Protocol

Given Orbital's API-driven development methodology[17] and web-ready nature it makes sense for the exchange of data with the Core to be made using a protocol which is already used across the world for the transmission of web traffic and which readily lends itself to the development of APIs. For this reason, Orbital will be built to use the HTTPS protocol to enable secure transmission of data between the Core and other applications. In addition, all communication between the Manager and the browser of an end user (where a user has opted to use Manger over a custom application) will be conducted over HTTPS.

HTTPS provides effective on-the-wire encryption and security for web traffic[7], and has a proven record of this. It is also supported in all major programming languages with little or no extension required, enabling rapid development and deployment of custom applications on the Orbital Core platform. When compared to the insecure HTTP protocol, HTTPS has a generally negligible impact on performance[8], especially when using modern computers, browsers and operating systems.

3.4.5 API Protocol

The orbital API protocol shall be RESTful, thus freeing it from additional complex requirements of maintaining stateful connectivity and sessions[9]. Adopting this RESTful pattern also encourages uniformity of design pattern, making the subsequent documentation and use of the APIs easier.

Since the RESTful API is using HTTPS as its transmission protocol it can take advantage of the various HTTP methods ("GET", "POST", "PUT" and "DELETE") to provide clear intent of any API call, reducing the possibility of accidental modification or deletion of data. In addition, the HTTP status codes allow for clear feedback to applications as to the state of their requests[10].

References

[1] [Online]. Available: https://github.com/lncd/Orbital-Core/wiki/Case-Studies

[2] Ccsds, "Reference Model for an Open Archival Information System (OAIS). Blue book," Tech. Rep. 1, January 2002. [Online]. Available: http://public.ccsds.org/publications/archive/650x0b1.pdf

[3] M. Day, *Digital Curation Manual.* DCC, UKOLN, November 2005, ch. Metadata. [Online]. Available: http://lncn.eu/bdi3

[4] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Principles behind the agile manifesto," 2001. [Online]. Available: http://agilemanifesto.org/principles.html

[5] N. Jackson. (2012, March) Data, Data Everywhere.... [Online]. Available: http://lncn.eu/cnz7

[6] B. J. Clark. (2009, August) NoSQL: If Only It Was That Easy. [Online]. Available: http://lncn.eu/jkv

[7] S. Thomas, *SSL & TLS essentials: securing the Web.* Wiley, 2000, ch. 1.3.

[8] A. Goldberg, R. Buff, and A. Schmitt, "A comparison of HTTP and HTTPS performance," 1998. [Online]. Available: http://lncn.eu/iprw

[9] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000. [Online]. Available: http://lncn.eu/dmxy

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC 2616, Hypertext Transfer Protocol – HTTP/1.1," 1999. [Online]. Available: http://www.rfc.net/rfc2616.html

[17] http://lncn.eu/jwx