Practical evaluation of SEEK and OpenBIS for biological data management in SynthSys; second report.

Eilidh Troup[2], Kevin Tomlinson[3], Kenton D'Mellow[3], Orlando Richards[3], Andrew J. Millar[1] and Tomasz Zielinski[1]

FINAL: 11 March 2016

[1] SynthSys and School of Biological Sciences, University of Edinburgh, Edinburgh EH9 3BF, UK

[2] EPCC, University of Edinburgh, Edinburgh EH9 3FD, UK

[3] Research Facilities Team, Information Services, University of Edinburgh, Edinburgh EH9 3FD, UK

Author contributions: ET and TZ modified and configured the software; KT provided servers and services; TZ and AJM wrote the report with input from all authors.

Funding: The work described here was supported by the School of Biological Sciences and the College of Science and Engineering, University of Edinburgh; and by BBSRC, EPSRC and MRC through award BB/M018040/1.

Acknowledgements: We gratefully acknowledge the Edinburgh Data Library for early input on DataShare (the University of Edinburgh's resource for data dissemination, based on DSpace).

Contents

I. Executive Summary

A. Objective

The objective of this joint project between University Information Services (IS) and the School of Biological Sciences (SBS) is to evaluate the provision of Biological Data Management systems and their integration with University Research Data Management solutions. The long-term aim is to comply with the University and Funder data mandates, while also adding value to ongoing research in SBS and the wider University. The benefits from streamlining data management would help to balance the School and user investment in establishing and adopting data management systems.

B. Background

We previously investigated the SEEK and OpenBIS data management systems and concluded that they provide useful and complementary functionality to support modern biological research (Troup et al., 2015[1], http://hdl.handle.net/1842/12236). However, technical aspects like provision of servers, system administration and integration with the University's infrastructure are more fitting on a School-wide level than in a project or Centre. Providing data management as a service aligns with the draft IS Research IT 10 Year Plan (Nov. 2015). IS has experience of providing biological software as a service, in the form of the Omero microscopy image database for MVM.

Here, we investigate similar models, in which IS and SBS divide the responsibilities for the physical infrastructure, system administration, software maintenance and customisation, and user support. This 2-month pilot project evaluated the potential of Synthsys' current systems, rather than undertaking actual provision. We sought to address the technical issues related to software hosting and to investigate options for integration of SEEK and OpenBIS with other university services.

C. Results

Both SEEK and OpenBIS can be hosted on IS infrastructure (virtual servers, data storage), subject to detailed agreement on the respective roles of IS and SBS in service provision. We report the requirements for each system. Integration with other University services was evaluated as follows:

EASE login (Cosign): successfully integrated with SEEK during the pilot, allowing single sign-on. OpenBIS: EASE adds security but single sign-on is not feasible, due to high costs and risks.

DataStore: successfully integrated during the pilot.

DataShare: integration technically possible but not completed. Simple prototype developed in SEEK; bugs in the SWORD communication client library are probably resolvable in reasonable time. OpenBIS integration will be possible but has not been prototyped.

DataVault: integration is not imminently necessary. The DataVault service is not fully defined but will have a degree of local control. Manual data deposit to DataVault will be possible, providing a sufficient exit strategy to retain data should SEEK/OpenBIS be shut down.

PURE: the desired integration is not technically possible, due to the lack of a write API in PURE. If alternative routes become possible, e.g. *via* DataShare, they will be worth testing.

[1] Troup, Eilidh; Clark, Ivan; Swain, Peter; Millar, Andrew John; Zielinski, Tomasz (2015) Practical evaluation of SEEK and OpenBIS for biological data management in SynthSys; first report. http://hdl.handle.net/1842/12236

D. Outcome

Technical evaluation: Enabling EASE login will enhance security on both systems, while DataStore provides safe storage within UoE. Exporting structured data sets directly to DataShare will save time in compliance for researchers and enhance the value of data over Supplementary Information files. However, it is impossible to generate the University-mandated metadata records in PURE, so the potential benefits from streamlining data management can be only partly realised.

Organisation of service: Two models are currently possible with IS (see Division of Responsibilities section). Both require SBS to support the systems at least in a pilot phase. A more SBS-based alternative might be preferable during the pilot, when requirements will change repeatedly, because it allows a more agile response to user needs. Other models should be considered in future, to gain value from scale and IS expertise while allowing local customisation and allocating costs appropriately.

II. Hosting and application stack

IS supports virtual machines running Scientific Linux 7 (SL7). IS provided a test machine (1 core, 4GB RAM, 50GB storage) and we were able to set up both repositories in this environment. IS also obtained SSL certificates/private keys required for the transport layer of OpenBIS, SEEK and Cosign.

A. OpenBIS

1. System description

OpenBIS is a Java based system with simple dependencies.

OpenBIS consists of two parts: a Web Server (WS) and a Data Storage Server (DSS), each listening on its own tcp port. The Web Server is a Jetty server running a pre-configured Java web application; it provides the primary user interface to OpenBIS. DSS is a standalone java application with embedded Jetty; it manages data storage and processing.

OpenBIS requires a Postgres database as its RDBMS.

The other component is OpenBIS Web Upload - an application developed at SynthSys for easy web upload of datasets and data pre-processing[1] (Troup et al. 2015; http://hdl.handle.net/1842/12236). It is a standalone java application with embedded Tomcat that listens on its own tcp port.

The data processing that can be triggered by OpenBIS requires python with various scientific libraries.

2. Installed dependencies and general setup

Java 8, Postgres and Apache were installed from default SL7 yum repositories (see Appendix A for names of packages)

The Anaconda Python distribution was chosen for its scientific libraries and it was installed in the context of a local user. This was an easier solution than manually compiling libraries using system scope packages.

OpenBIS was installed using the installer provided. IS-provided SSL certificates had to be converted and imported into a Java keystore to make them usable by OpenBIS.

Apache and OpenBIS were configured so that OpenBIS can run behind a reverse proxy to allow one point of access (no port numbers). Synthsys's OpenBIS Web Upload did not work correctly behind the proxy, but as it is an in-house built application this could be easily rectified in the future.

B. SEEK4Science

1. System description

SEEK is a Ruby on Rails web application and has multiple system dependencies.

Unlike Java based applications SEEK does not ship with all its dependencies but uses the gem package management system to build them directly on the server. Unfortunately such building is still dependent on various linux system libraries. SEEK installation is documented only for Ubuntu and matching SL7 libraries had to be found.

The main part of SEEK is a Ruby on Rails web application, which requires a ruby server. Apart from the main SEEK, 3 additional services have to be running:

- background jobs: the ruby program responsible for background handling of the data and metadata (for example triggers document indexing)
- solr service: text indexing engine
- libreoffice demon: for handling pdfs and MS Office documents

SEEK also requires MySQL database as its RDBMS and Virtuoso DB for storing knowledge graphs.

2. Installed dependencies and general setup

Over 30 packages were installed from default SL7 yum repositories (see Appendix B for names of packages).

MySQL is not supported on SL7 so MariaDB, its in-place replacement was used instead. A custom yum repository had to be defined (see Appendix C) to allow access to the recent version of MariaDB.

Virtuoso DB was installed using yum, but it lacked the Conductor web application required for user management and web sparql queries. This had to be added manually (see Appendix D).

A ruby web server is necessary to run SEEK. We used the Phusion Passenger module for Apache. SL7 yum repositories contained old version of Passenger, so instead we installed the most recent version using ruby gem installer in the context of the local user.

The actual SEEK was installed following the procedure from its documentation. It involved downloading the source from GitHub and building it locally. Next, the Virtual Host in Apache was configured to use the Passenger module to run the ruby applications. For the 3 additional services, startup scripts were created following the SEEK documentation and placed in `init.d`.

III. Integration with the University's Services

A. Cosign

Cosign is Edinburgh University's single point login mechanism which allows user access to various web applications using their university credentials.

IS installed and configured the Cosign module for Apache on the pilot machine and registered it with the University's Cosign service. On its own, it already provides an extra layer of security, as access to

SEEK or OpenBIS can be restricted to Cosign logged in users (provided that OpenBIS runs behind Apache reverse proxy as in the tested setup).

Apache's cosign module sets the REMOTE_USER variable to the login of the request-making user. SEEK runs via apache Passenger module and this variable is visible to the Ruby code. We implemented a new authorization mechanism in SEEK, which uses the remote user value to authorize users. It is a complete and fully functional solution that integrates SEEK with University Cosign infrastructure.

This SEEK extension could potentially be further developed to integrate with the University's LDAP service in order to populate user profiles with information stored there. However, as the profile creation is a one-off operation the costs of the work involved do not justify the potential gain.

Http requests to OpenBIS are handled by its own embedded Jetty server. As a result, apache's REMOTE_USER variable is not accessible to it. We overcame this obstacle by using the Apache rewrites module to add a special http header which contains the remote user name. We could access this request variable from within test servlet that we deployed on the OpenBIS embedded server. This cannot be a long term solution as it creates vulnerability to headers hijacking. The proper Cosign filter must be implemented which could be based on an existing filter for Tomcat.

However, the main obstacle in Cosign integration lies within the OpenBIS application itself.

The available extension points to the authorization mechanism are based on implementing the `IAuthenticationService` interface with the method:
`authenticateUser(String user, String password).`
At this point of code execution, only the username and password extracted from the login form are accessible and not the original http request. The original full request is necessary to access the REMOTE_USER header/variable set by apache/jetty Cosign filter.

Hence, modification of the upstream authorization mechanism will be necessary. But that is based on Spring framework security which extensively uses automatic dependency injections and aspect programing which makes it difficult to change or even identify the classes that require modification. Moreover, we do not have access to the build procedure for OpenBIS or even a concrete version of the source code[1] (see http://hdl.handle.net/1842/12236). As a result, hand picking and replacing binary artefacts with OpenBIS war file would be necessary.

We conclude that adding Cosign authorization to OpenBIS is a high risk, low gain operation. However, we still recommend using Cosign in front of the reverse proxy in order to add extra layer of security even if Cosign user information will not be used by OpenBIS itself.

B. DataStore

DataStore is a University service for storage of active research data, which assures data durability and recovery options. It complies with funding bodies polices on Research Data Management and as such should be used to store all research output. For that reason, it is important to integrate data repositories with this system.

IS provided and configured a 1TB storage area on DataStore. It was mounted on the test machine and both OpenBIS and SEEK were setup in such a way that all of the user uploaded files are stored on DataStore. At the same time, all the system and application files remained stored locally on the server to assure good performance.

We did not address permissions issues of the storage area that would assure restricted access to its content. To achieve this the uid/guid must be mapped on the server using centralised UID and GID spaces. This can be achieved with centrally provisioned University functional accounts and a corresponding locally mapped user on each host accessing the storage. This can be achieved with local accounts or by hooking into a central user directory like Active Directory using, for example, SSSD.

C. DataShare

DataShare is an Edinburgh University service for preserving and making publicly available research data, maintained by the Data Library (www.datashare.is.ed.ac.uk). The University has designated DataShare as a service for providing 10 year access to datasets that support published papers, which is required by funding bodies.

Although SEEK and OpenBIS offer extra value to the researchers and even may be better suited for sharing data within a given community, we cannot guarantee 10-year access on those platforms (for example, due to discontinuation and obsolescence of the software). Only the institution can provide this level of guarantee.

We envision SEEK and OpenBIS being in daily use by researchers, hosting multiple versions of data and analysis results, whereas DataShare is the final destination for static, 'golden copies' of published data sets. Researchers therefore require easy migration of data sets to DataShare.

The Data Library has agreed to automatic content deposition to DataShare using the SWORD protocol (http://swordapp.org/about), which makes it technically possible to implement data set migration from the repositories to DataShare.

The Data Library provided us with a definition of the metadata required for data set deposition in DataShare. As this kind of information is not typically included in SEEK or OpenBIS asset descriptions, a new UI form must be implemented to gather those details. Once deposition details are gathered, relevant files will be assembled and packed in a zip archive container to be submitted to DataShare using the Sword client.

We have access to the SEEK source code and build procedure so we can add new functionality to it. However, as we are not proficient Ruby programmers and this is a substantial modification of the code (new model objects, new relationships, new background tasks) we believed it the most risky part of the pilot, and concentrated our effort there.

We developed a skeleton UI for dataset deposition, leveraged the SEEK background jobs mechanism and implemented a dataset export procedure. With all those elements ready, we would have had a simple working solution to DataShare integration. Unfortunately, the existing Sword client in Ruby could not communicate correctly with a Sword-capable server (we used DSpace for testing, which is the software on which DataShare is built). We fixed a few bugs in the client that had prevented it from retrieving service contact from the server. We could then upload a data file to the server but we still had difficulty in transmitting all the metadata required by the Data Library.

The main issue is that including the Sword Ruby client in the SEEK dependencies (gem list) breaks SEEK's RDF generation subsystem. The error is caused by an unknown conflict in the underlying libraries and at this point we don't know how to resolve it.

We don't have access to the OpenBIS build procedure, so we cannot modify its UI to provide a DataShare deposition form nor add a migration procedure. However, OpenBIS provides an API,

which we are already using in our OpenBIS Web Upload tool. It will be possible to extend our tool to include the export functionality and use it as the front end for dataset migration. We could use the existing Sword client for Java, which we also tested with the DSpace server.

Integration with DataShare is probably the most important from the user perspective so it is encouraging that it is technically possible and that we were able to make a fair advance in implementing it in a short time in the pilot. Data migration to DataShare will also benefit from Cosign integration, as the UUID is necessary for data upload.

D. DataVault

DataVault is a University service that offers long term preservation of non-active data. The service interfaces are still under development, and are expected to be finalised by late 2016. There is no imminent need for integration of the biological repository with this service. However, as we cannot guarantee the 10 years of service that is required by current policies, we propose an exit strategy in case of discontinuity of the SEEK or OpenBIS, involving archival to DataVault of the entire service and all datasets contained within.

All the datasets cannot simply be migrated to DataShare as it only stores publicly accessible assets, so another solution is necessary. However, we could re-use the part of the code that assembles the data in one container for DataShare upload, and use such packages as storage units in DataVault.

The actual data transfer would be implemented as a manual deposit into DataVault with support from the IS Research Data Management support team.

E. PURE

PURE is the University's current Research Information System which is the main point of reporting research output. It provides information on generated data sets, so integration with this system would be of high value as it would reduce researchers' administrative burden.

We investigated with IS Applications Division to establish possible integration paths. It turned out, at this point, that integration with PURE is not technically possible.

PURE is a commercial product that lacks a write API, so external software cannot create records in it. The alternative, directly modifying PURE's backend RDBMS, is not sustainable from a maintenance point of view and it is too big a job to be performed using SBS resources.

The Data Library is investigating a semi-automatic batch upload of its records to PURE, so exporting data sets to DataShare might then be enough to make them accessible in PURE. If this batch upload was successful, we could also potentially replicate the process, in order to access PURE without going *via* DataShare. Until the details of this process are established, it cannot be evaluated. We conclude that integration of any relevant biological data management system with PURE is not currently feasible.

IV. Division of responsibilities for long-term hosting

Going forwards, Information Services currently offers two models. First, IS can continue to provide the virtualised infrastructure on which SynthSys can host both services, together with standard support for the group's use and configuration of this infrastructure. IS could also provide additional consultation effort above and beyond this at standard IS consultancy rates to further develop the service. However, IS cannot offer systems administration for the servers: SBS would need to provide.

Second, further work would be required before IS would be able to take over the provision of the service itself. In particular, the application development would need to be completed, and ideally incorporated in the upstream code base (i.e. become part of the SEEK/OpenBIS core) to ensure ongoing sustainability. Ongoing development (and any other operational/support tasks which require direct database and system access) would not be possible directly on the live service, but development and test environments could be created to allow for ongoing collaborative development. Commitment of sufficient infrastructure and staff resource to support the service would be required, which would typically require either external funding to recruit or reassign support staff, or reassigning existing staff effort based on a business requirement to provide the service (which itself will typically be best justified by a demonstrably wide user base).

SynthSys' immediate concern is to provide sufficient, relevant services to test user uptake in a pilot phase, where the second model is inappropriate. Hosting entirely within SBS is a third model that might be more appropriate in a pilot, when requirements will change repeatedly, because it could allow a more agile response to user needs. In future, other models with shared responsibility could to gain value from scale and IS expertise while allowing local customisation, support, and appropriate allocation of costs. One such model would be for IS to fund a developer/system manager within SBS.

V. Concerns for long term hosting

A. Security Concerns

Like any server-side software both repositories can potentially contain vulnerabilities that could be exploited to compromise the system. They are developed by small academic groups, so threat detection and mitigation can take longer than in the case of popular, commercial products. But for the same reason, there is small probability of dedicated attacks on either system, the main concern should rather be vulnerabilities in the libraries/framework used.

a) OpenBIS

In case of OpenBIS, the higher risk is the Jetty server that is packed/embedded within the WS and DSS components. Because Jetty is part of the OpenBIS distribution it cannot be simply updated every time a new release of Jetty is available but has to be linked with an OpenBIS update.

Proposed mitigations are: running OpenBIS behind an Apache reverse proxy and Cosign filter; proper configuration of Java Security Manger using Policy Files and; running OpenBIS as a local service with limited access to resources.

b) SEEK

The entry point to SEEK is by the Passenger module for apache. As SL7 lags behind the official release of Passenger, this module was installed using ruby, not yum repositories. Another risk factor is the sheer number of system dependencies (see Appendix B), which may delay library updates or make them difficult.

Again, using a Cosign filter in front of SEEK reduces the risk of malicious requests to the system.

c) Network access

Typically network access to the server is limited to port 80 for http and routed from there by apache proxy. However, both tools are still being actively developed by their authors or us, and as we discovered, we often need access to backend databases or to directly call service endpoints in order

to diagnose and fix problems. For that reason, SBS developer/admin requires network access to the following services/ports:

- MySQL,
- Postgres,
- Conductor (Web Interface for Virtuoso)
- 3 ports on which OpenBIS components run
- 1 additional diagnostic port (for example to run SEEK as standalone)

As such setup was against IS standard practice, the pilot server was outside their network. For long term hosting the issue with opening additional ports could be alleviated by the use of a test environment allowing as much open access as necessary. Production and live service hosts would be entirely restricted behind various combinations of router ACL's, firewall rules and load balancer configurations. Direct access should not be necessary or possible to the service components from outwith the service or management environment.

In the model where IS provide only the infrastructure for SBS to run the service themselves then the hosts can have whatever firewall rules / access mechanisms are required by SBS as they would sit on network segments isolated from production IS service traffic.

B. Maintenance

IS is using Saltstack to centrally manage server configurations. In the model where IS provide only the infrastructure for SBS to run the service themselves the management and deployment can be carried out in whatever fashion SBS see fit. IS would not manage the configuration of these hosts beyond their initial basic operating system deployment.

OpenBIS has limited system dependencies and as a result the responsibilities of IS and SBS parties can be easily separated. Server updates can be automatically managed by IS sys-admin, while, OpenBIS update could be independently performed by an SBS admin/developer.

In the case of SEEK, SEEK updates most likely will require changes in server configuration so involvement of IS will be necessary.

C. Backup and mirroring

Integration with DataStore provides backup to the uploaded data files. However, it does not cover content of RDBMs used, which will need to be backed up separately.

If the repositories are meant to be used on the school level, 24/7 access to the data should be guaranteed. This could be achieved by mirroring/load balancing which has not been investigated as part of this project, and so further work would be required here. There are many potential solutions but the underlying applications would guide what is possible by the traffic flows, data distribution and segmentation models they can support. In the model where IS provide only the infrastructure for SBS to run the service themselves, SBS can investigate and implement any solutions as appropriate. IS can provide some help and guidance on potential solutions, with the understanding that no immediate solution is understood.

The infrastructure provided by IS will be via virtual machines running on the Vmware platform and so will offer backups, Vmware DRS , Vmware Vmotion, Vmware Svmotion and Vmware High Availability for additional protection and resilience of guests and their storage.

VI. Estimated Costs

A. Infrastructure costs

We suggest 3 servers (for SEEK, OpenBIS, and RDBMS) each with 4 cores, 32GB RAM, 50GB storage (200GB for RDBMS). Cost for SBS provision with IS hosting on virtual machines:

£957 per year per guest (£2871 per year)

£50.65 per year for additional 200GB RDBMS storage.

£65.80 per year for backup.

DataStore costs are £175 per TB per year including backup.

Cost for IS service provision rather than IS hosting is the same.

SBS hosting would require hardware servers, which could initially be the existing SynthSys hardware; new hardware would require setup by an SBS sysadmin.

B. Maintenance time

Costs can only be estimated for the first model, SBS provision with IS hosting. The maintenance costs for IS provision of the service cannot be estimated accurately without understanding the scale of the user base.

Initial setup:

IS: 3 days of FT. SBS: 1 week of FT for the software manager

Ongoing updates:

IS: Included in infrastructure hosting costs. SBS: 2 days per month for the software manager.

Either IS or SBS hosting would entail systems administration of the servers. IS hosting does not include sysadmin functions. Routine server maintenance, security and recovery for one server would take only 1-2 days per month, if staff time with the right expertise is available in SBS. This issue was also noted previously[1].

VII. Conclusions

Integration of SEEK and OpenBIS with most of the IS services tested is technically possible and offers multiple advantages, which are available whether SBS or IS hosts the data management systems. The desired integration with the current PURE is technically impossible. Service provision by SBS is currently the only option, because service provision by IS requires a large, probably cross-School user base. Such a user base could only be developed by providing the live services, and would require at least a medium-term commitment and/or a clear exit strategy.

Provision for Electronic Lab Notebooks, which SynthSys is now evaluating, faces similar issues and potential for integration with IS. ELNs have a greater range of commercial providers. Several SBS groups are heavily committed to customised wiki ELNs on the IS Confluence system. We find analogous needs, for local customisation to add immediate value, and for read/write APIs to allow integration with upstream and downstream systems.

VIII. Appendices

A. Appendix A – OpenBIS yum dependencies

Installed yum packages:

```
java-1.8.0-openjdk
postgresql-server
httpd
httpd-devel
```

B. Appendix B – SEEK yum dependencies

Installed yum packages:

```
httpd
httpd-devel
apr-devel
apr-util-devel
java-1.8.0-openjdk*
wget curl mercurial ruby libssl-dev
readline-devel
libxml++*
yum nodejs
sqlite-devel
libcurl-devel
poppler-utils
libreoffice
libreoffice-headless
ImageMagick
ImageMagick-c++-devel
libxslt-devel
libpqxx-devel
libtool
gawk
libyaml-devel
autoconf
gdbm-devel
ncurses-devel
automake
bison
libffi-devel
mysql-devel mysql-libs
openssl-devel
virtuoso-opensource
virtuoso-opensource-utils
```

C. Appendix C – MariaDB installation (MySQL replacement)

```
1. Definition of MariaDB repository
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.0/rhel7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

2. Installation of MariaDB

```
yum install MariaDB-server MariaDB-client
mysql_upgrade -p
/usr/bin/mysql_secure_installation
```

D. Appendix D – Installation of Conductor - Virtuoso extension

```
# start virtuoso (from etc/virtuoso)
virtuoso-t &

# Install conductor
# download conductor (Version 6.1) from
http://s3.amazonaws.com/opldownload/uda/vad-
packages/6.1/virtuoso/conductor_dav.vad

# copy it to the virtuoso running folder, etc/virtuoso
# login to virtuoso (if needed default password is dba)

isql-vt -U dba
vad_install ('conductor_dav.vad', 0);

# conductor is under works
# http://host:8890/conductor
```