

Repository as a Service (RaaS)

Stuart Lewis - The University of Auckland Library, New Zealand: s.lewis@auckland.ac.nz

Kim Shepherd - The University of Auckland Library, New Zealand: k.shepherd@auckland.ac.nz

Yin Yin Latt - The University of Auckland Library, New Zealand: y.latt@auckland.ac.nz

Andrea Schweer - Library Consortium of New Zealand: schweer@waikato.ac.nz

Adam Field - EPrints Services, University of Southampton, UK: af05v@ecs.soton.ac.uk

Introduction

In his oft-quoted seminal paper '*Institutional Repositories: Essential Infrastructure For Scholarship In The Digital Age*' Clifford Lynch (2003) described the Institutional Repository as "a set of services that a university offers to the members of its community for the management and dissemination of digital materials created by the institution and its community members." This paper seeks instead to define the repository service at a more primitive level, without the specialism of being an 'Institutional Repository', and looks at how it can be viewed as providing a service within appropriate boundaries, and what that could mean for the future development of repositories, our expectations of what repositories should be, and how they could fit into the set of services required to deliver an Institutional Repository service as described by Lynch.

Background

This paper is a fuller description of the submission by the authors to the Open Repositories 2011 'Developer Challenge' in which contestants were asked to 'Show us the future of repositories'. The team was made up of staff from the University of Auckland Library, EPrints Services, and the Library Consortium of New Zealand. The authors had not previously anticipated working together, however during discussions in the 'Developers Lounge' it became clear that there was overlap in not only their ideas to be presented, but also in some of their prototypes to be demonstrated.

The core ideas of the team centred around two concepts: *Expectations* and *Interoperability*. The former concerns the expectations of repository software that exist within the Open Repositories community, and the latter with interoperability between repositories and other supporting systems. The two ideas played together to form the basis of the competition submission. The authors believe that the future of repositories will change when we revise our expectations of today's repositories, and once the plumbing required for effective interoperability is in place.

These two ideas come together to form the notion of 'Repository as a Service'.

... as a Service

Apparently first coined in public at a conference in 2005¹, the phrase Software as a Service is used to describe the online hosting of software for the use of others. Rather than running the software locally, users / institutions / corporations are able to access the hosted software as a service (for free, or via payment). The benefits of this model include outsourced support, no hardware to procure or run, no software to install or update, and the ability to access the software from any Internet-enabled computer.

Following the widespread adoption of the phrase, other 'as a Service' models have been coined, including:

- **Infrastructure as a Service:** This includes low level compute facilities and raw disk storage. Usually provided by large hosting companies such as Amazon AWS and Rackspace, the time required to buy a server with configurable resources, or to get access to terabytes of disk changes from weeks or months for an internal procurement, to a matter of minutes from a supplier of Infrastructure as a Service.
- **Platform as a Service:** The provision of a 'stack' of components that can be used to put together a solution, and available instantly online, is known as Platform as a Service. These often sit on top of the infrastructure, and hide the complexities of running servers, storage, or core components such as a database or authentication system. Instead, users are able to more easily build or deploy solutions that make use of features of the platform.

All of the 'as a Service' systems can be typically included in the term 'cloud computing' where the commodity or service (software / infrastructure / platform) is provided by an external provider who takes the responsibility to provide new or scaled-up instances of the service as demanded by the subscriber. In our notion of Repository as a Service, we ignore the question of who provides the service, but look instead at the characteristics of the systems provided 'as a Service'.

The Open Repositories Landscape

The current landscape within the Open Repositories community is currently dominated by tools such as DSpace and EPrints. These tools have developed over time to provide complete turn-key solutions to creating a repository. As such they provide item description and storage mechanisms, ingest facilities, curation tools, and dissemination methods.

Because these repository tools come complete with all the functionality required to run a repository, the task of commissioning a new repository has become an easy task: install the software, perform some configuration, accept deposits. When first installed, they are typically configured to perform the role of a normal institutional repository storing pre- and post-prints, electronic theses, and other similar document types. It is then possible to configure these repository platforms by adjusting different settings such as the look and feel (to match institutional branding requirements), metadata input forms, metadata field labels, accepted file

¹http://en.wikipedia.org/wiki/Software_as_a_service#History

format types etc.

Because of the relative ease with which this type of repository can be installed, there is usually no need for any input from developers, other than perhaps during the initial development phases where a few small features may be needed. Therefore for most installations, there is no available resource to make substantive changes to the software if required, meaning that only the feature set available out-the-box is available. Whilst these repository platforms have grown over time to meet the majority of requirements to run an 'institutional repository', many users do not have the resources available to make any changes required to allow the repository to be used in others ways. The implications of this are described later.

Expectations

As implementations of repositories are maturing and becoming seen as less of an innovation and more as a component of standard digital infrastructure, stakeholders are expecting the repositories to keep up to date with the modern web environment and to provide best-of-breed implementations of each part of the software. Examples include feature-rich web interfaces with features to allow social interaction with the contents, powerful search and discovery mechanisms, and deposit interfaces that integrate with numerous external sources to pre-populate metadata or text mine data from the deposited document.

As well as expectations of what facilities a repository should provide, expectations have grown around the collection policies of the typical institutional repository. In contrast to the growing expectation that the repository software should do more and more, the expectation of what goes into a repository has not changed. Repository collection policies often restrict themselves to 'research outputs', that is, the outputs of research, such as papers, reports, or theses. Some repositories are used for images and small video collections, and some are starting to be used for research datasets.

Therefore the overall trend seems to be one of increasing expectations of our repositories in the area of functionality, but only a small, or no change, in our expectations of the uses we put them to and what we store in them. It seems that we want them to manage the same content, but with more bells and whistles.

We propose that we should change our expectations and shift the focus to the service provided with less emphasis on the bells and whistles. We shall now explain how, and why.

Examples

We shall now give two examples of innovative uses of traditional repository software. They are not innovative in what they do, but they are innovative in their use of a repository to achieve this function. By keeping to our current expectations of what can be stored in a repository, we will fail to exploit the potential they give us in managing, preserving, describing, and giving access to many other types of content such as these

SuperIndex / Skylight

The University of Auckland Library has many locally created collections of specific bibliographic records and digital content. They are using different systems such as proprietary web enabled database systems, WordPress sites, and Microsoft SQL Server-powered web sites. Over the years, the growth of the databases and diversity of platforms has led to difficulties in upgrading and maintaining all the different systems to the latest versions of their respective platforms. For custom-built systems this is even harder when the system was built many years ago with resources from a specific grant. Therefore we have started the SuperIndex project.

The project has three main requirements,

1. to consolidate all the collections into one central repository;
2. to provide each collection with their own branded and addressed public interface;
3. to allow each collection to be discovered and indexed by external systems;

To consolidate all the collections into one central repository, we chose DSpace which is used by over 1,000 organizations across the world to manage their digital assets. This decision was also based on the availability of local staff with extensive DSpace expertise. Based on the project requirements, we have setup a very vanilla DSpace as a management tool. The main difference to most DSpace repositories is that the DSpace native web interface is only used by internal collection administrators, not external users.

To allow each collection to have their own branded public interface, the local Library Digital Development Team developed a new open source collection viewer tool called Skylight (<http://skylightui.org/>). Skylight is developed using the CodeIgniter PHP framework which is based on the Model-View-Controller development pattern. This has allowed rapid development of functionality and easier maintenance because unlike in a traditional repository user interface, no consideration needs to be paid to how the rest of the repository works and and the associated application item integrity implications, as all content is open access and read-only.

Each instance of Skylight is able to expose a collection from DSpace using a unique domain, such as {collection}.auckland.ac.nz. This also allows the DSpace platform to only provide the repository service, whilst Skylight provides the user interface which can be highly branded and customised to best fit each collection's content, metadata, and style requirements.

The lessons to learn from this example are:

1. The end-user interface for the collection does not need to be provided by the repository software. Doing so would have been difficult, and it would not have been possible to provide the highly customised sites for each collection that Skylight is able to. Without the use of Skylight, each collection would require its own repository installation.
2. Repositories can be used as collection management tools. They don't have to be restricted to the typical uses of an institutional repository such as theses, articles, reports, and images. Anything that is made up of structured data, with or without digital objects, can be stored in a repository. This allows us to use repositories for collections where we might normally have used other technologies - by doing so, we reduce the number of systems that we provide and maintain.

Tweet archive

A less obvious example, created specifically for the Open Repositories 2011 Developer Challenge is an archive of all tweets that included the #or11 hashtag. Tweets are not typically seen as bibliographic records, but they are simple text objects described by a basic and familiar set of structured data: id, author, date and subject (hashtag). They also reference related resources by linking to images and other content on the web. It was felt that #or11 tweets would be the perfect “atypical, yet relevant” resource that we could harvest and archive to show the potential flexibility of existing repository services.

Tweets were harvested from 3-4 weeks before Open Repositories 2011 and numbered in the thousands by the time the conference was ending.

In keeping with the philosophy about expectations noted above, the repositories involved in this example were expected only to archive and curate content, and expose that content in a standard way, with the end-user UI provided by Skylight. Content was exposed using standard protocols such as OAI-PMH, and web services like Apache Solr; Solr was chosen in this particular case because we were integrating with a search-focused web user interface, not another system. The DSpace curation system was demonstrated by writing a translation service using the Microsoft Translation API that translated tweets into Korean as they were harvested, and stored the translated text in addition to the original.

The tweet archives were implemented in both DSpace and EPrints, independently of each other. Solr support was added to EPrints during the Developer Lounge sessions to enable integration with Skylight. During the Developer Challenge presentation, the presenters were able to switch between these two sites seamlessly. In addition to faceted search, related item suggestion and other standard Skylight features, the front-end sites were customised to include an interactive timeline widget hard-coded with the conference schedule: users could browse tweets that were sent during a particular session or event.

Repository as a Service (RaaS)

Our proposed notion of ‘Repository as a Service’ could be seen as a particular Platform as a Service. The platform that is being offered is that of a repository - a system that can store and describe digital objects, whilst ensuring they are curated and managed appropriately. By thinking of the repository not as a traditional Institutional Repository and all the baggage that is associated with that (full rich user interfaces, configurable deposit interfaces, etc) but as a service, we can imagine new uses for the storage, description, and management of objects, such as the examples above. We need to ensure that we do not self-impose limitations on the repository platforms by restricting our personal expectations of repositories.

A repository being provided ‘as a Service’ will need to have the following characteristics:

Easy provisioning

A typical characteristic of ‘as a Service’ products, is that from the user’s perspective the

provisioning is easy, immediate, requires no other human intervention, and is cheap (an initial empty repository is cheap in the sense that it uses little resources). The user need not know how the service was provisioned, where it was provisioned, or where it might run in the future. The key is that it was provisioned easily, under the ownership of the person who decided to create a repository.

Immediate provisioning is particularly important for some potential uses of the repository such as the tweet archive example above. If a new world event occurs for which the hash-tagged tweets need to be captured, immediate provisioning (without the need for a committee decision) is required in order to get the repository in place quickly to start gathering the data. For the typical repository service today, this means that the ability for a user to create either a whole blank new repository, or a new collection within an existing repository service, needs to occur with very few key-presses and no 'approval' mechanism from the library or other system owner.

Scalability

With Repository as a Service provisioned by a potential user and no human intervention means that scalability has to be inherent to the repository, as the size of the repository is unknown at the point of conception. From a resource point of view, especially in the age of cloud computing, this is less of a problem, however repository platforms will need to learn how to scale, perhaps by learning lessons from other infrastructure technologies, for example 'sharding' where the system is split up into chunks which are spread across different systems but combined together for presentation to the user.

Resource accounting

As already stated, an empty repository should cost very little, or nothing, as it is using no resources. This view is the opposite to a typical repository installation which is seen to require hardware and resources. As the repository grows, it will be important to account for resources used in order to bill the user appropriately. This paper will not discuss who should fund the repository service (the institution, the library, the researcher, the IT department, the research grant), but a scalable repository will have the ability to grow large, and a large repository will cost more money in its growing use of resources. Better resource accounting (typically in terms of data stored + data transferred + work done) will allow better proportioning of the cost of the service.

Interoperability

Another characteristic of a service is the ability for it to be interchanged to another service provider or system. A crude example of this would be ability to swap an EPrints repository for a DSpace repository, with the user seeing no difference to their user interfaces. In order to achieve this, high levels of interoperability are required, both at current integration points, and at lower levels.

Currently most repositories support interoperability at the point of deposit by using SWORD, or at the point of external discovery by using OAI-PMH. However not all repositories support integration-points such as APIs or other services required to provide external user interfaces.

For the repository to become a black box swappable-service, we need to better support the modularisation of repositories in order to allow different interfaces to be built in and out of them. By doing so, the repository service in the middle can be swapped, perhaps for reasons of scalability, changing environments, or new requirements.

Decommissioning

If it is easy to create a repository, it should be easy to discard one. Many services offer a '30 day free trial', maybe repositories should do the same? A repository may also only be needed for a while, for example to collect a data set. Once that data set has been collected as a collection of repository items, the collection may then be extracted, re-worked, and re-deposited as a single combined dataset. Therefore there should be an expectation that repositories can be transient tools as well as permanent archives.

Some repositories may however need to be archived at the point of decommissioning. Perhaps repositories need the capability of archiving whole repositories in order to create these 'repositories of repositories'?

Quality of Service

A final characteristic of many 'as a Service' systems is the quality of service they provide. This is manifested in two ways:

1. Because the system is built to scale and to operate with minimal central support intervention, it is likely that a more robust and reliable service will be built compared to a single repository running on a single machine. If a repository service is to be used in the day to day activities of researchers as they collect and work with data, then the service must be run with the same levels of expectations regarding uptime and integrity as other enterprise systems such as email, filestore, or DNS.
2. With immediate provisioning, scalability, and no requirement for central support, the quality of the service in terms of ease of use, instructional materials, and options for configuration must be high. A user should be able to use the repository with little or no help.

Conclusions

At present we tend to look at our repositories as single software systems that provide a complete end-to-end repository service. In terms of an institutional repository, we have an idea about what this means: what we can store, when it gets stored, who can deposit. If we change our views of the repository to make the repository a 'service' rather than a single 'system', we may find new uses for the repository that we had not previously considered it for. If we continue to restrict our repositories to 'research outputs', then we may miss out on allowing our repositories to provide a valuable service to users during the research process. Our repositories can serve many more uses than we are currently allowing or expecting them to.

In order to support this however, we need to change our expectations of the repository. It

is unlikely that a single large piece of software will ever provide all of the functions required to support every user, every use case, and to deliver the characteristics of a 'as a Service' system. Perhaps, instead of asking our repository systems to support an ever-increasing array of functions within their native user interfaces, we need to encourage them to support new and richer integration points allowing us to build new tools that interact with the repository.

As our expectations of repositories grow, and as their ability to provide a 'Repository as a Service' improve, we will talk less about what repository software we run and how many full-text items it holds, and instead talk about service we are providing, the wide range of new item types we are taking care of, the latest deposit interfaces that our users have built to get their data into our service, and the new user interfaces that have been built on top of it.

Have high expectations of the repository service you could provide, but don't expect your current single repository software to fulfill all those needs.

Acknowledgements

The authors wish to express their thanks to the organizers of the Open Repositories conference, the DevCSI initiative staff who ran the Developers Challenge, and to the JISC and Microsoft for sponsoring the competition.

References

Lynch, Clifford A. (2003) "Institutional Repositories: Essential Infrastructure for Scholarship in the Digital Age". *ARL*, no. 226 1-7. <http://www.arl.org/resources/pubs/br/br226/br226ir.shtml>.