The DSpace Open Source Digital Asset Management System: Challenges and Opportunities

Conference Paper · October 2005

DOI: 10.1007/11551362_22 · Source: OAI

3 authors, including:

Mackenzie Smith
University of California, Davis
31 PUBLICATIONS **676** CITATIONS

SEE PROFILE

The DSpace Open Source Digital Asset Management System: Challenges and Opportunities

Robert Tansley[1], MacKenzie Smith[2], and Julie Harford Walker[2]

[1] Hewlett-Packard Laboratories, 1 Cambridge Center, Cambridge, MA 02142
robert.tansley@hp.com
[2] MIT Libraries, 77 Massachusetts Ave, Cambridge, MA 02139
kenzie, jhwalker@mit.edu

Abstract. Last year at the ECDL 2004 conference, we reported some initial progress and experiences developing DSpace as an open source community-driven project [8], particularly as seen from an institutional manager's viewpoint. We also described some challenges and issues. This paper describes the progress in addressing some of those issues, and developments in the DSpace open source community. We go into detail about the processes and infrastructure we have developed around the DSpace code base, in the hope that this will be useful to other projects and organisations exploring the possibilities of becoming involved in or transitioning to open source development of digital library software. Some new challenges the DSpace community faces, particularly in the area of addressing required system architecture changes, are introduced. We also describe some exciting new possibilities that open source development brings to our community.

1 Introduction

DSpace is a digital asset management system, most commonly used as an institutional repository system by research universities. The system was initially developed by a joint HP and MIT Libraries development team. Since its release in November 2002, it has achieved widespread adoption; at the time of writing there are well over 100 DSpace instances running distributed among all of the continents in the world.

Both HP and MIT Libraries had the same original goal of building DSpace to discover what it takes to build and deploy a repository system to capture, preserve and disseminate an organisation's born-digital assets. It was clear from the start that this goal was not unique to HP and MIT Libraries, and that there are several compelling reasons for pursuing an open source development model:

- It was clearly not sustainable for HP and MIT Libraries to support the entire community of DSpace users. An open source model allows each organisation to customise the platform for their own particular local requirements, and to enable these customisations to be shared as appropriate;

A. Rauber et al. (Eds.): ECDL 2005, LNCS 3652, pp. 242–253, 2005.

- It is a focus around which a community of researchers and practioners can form, exploring the areas of managing digital content and long-term preservation of digital materal;
- A wider group of stakeholders and developers who understand the system ensures the longevity of the system and content stored within instances;
- It allows researchers, practioners and developers to work together rather than as islands or silos;
- It provides those groups an opportunity to see the fruits of their work adopted and deployed by end users.

This paper describes how we have made the transition from regular co-located team-based development to a broader open source development model, in which the software source code is maintained and developed by the community around it as a whole, as opposed to being principally worked on by one particular team or organisation. We also describe some of the challenges and opportunities this brave new world presents.

2 The Transition to Open Source

The initial period of DSpace software development before the release of version 1.0 in November 2002 was a fairly typical software project. A co-located HP and MIT team of library staff and developers specified, designed and built a 'breadth-first' system, which, although greatly informed and assisted by members of MIT Libraries staff and MIT faculty 'early adopters', was a largely closed process.

Over the year following the release of version 1.0, the main change from the initial development period was that a great deal of feedback in terms of bug reports and functionality was received from a widening group of users. However, DSpace software development and maintenance was still a centralised process performed by the HP and MIT Libraries team; hence, although the source code was open and under an open source license, this was not really an open source development model. Although at the time this did attract some criticism, this is understandable and reasonable; one cannot simply remove access control on the source code for allcomers. From a very early stage, many universities libraries' reputations and patrons were relying on the stability of the DSpace software.

The first signs that a real technical community was forming around DSpace appeared around technical support for installation and configuration problems. In the first days after the 1.0 release of the DSpace software, members of the HP/MIT development team had to answer every technical query in this area, or leave a question unanswered, which would soon lead to frustration. It was only a matter of a few months before many questions were being answered by members of the DSpace community outside of the HP/MIT team. This was very encouraging, as it demonstrated a willingness by members of the community to donate time and effort to help each other out. This 'self-help' technical support model is probably where the average DSpace user first encounters the difference between an open source system and a commercial system, and accordingly was the first area where the DSpace community really started to gel.

However, at the same time, the HP/MIT team realised that more needed to be done to support a move to real open source development, where development and maintenance of the code itself is owned by the community as a whole.

One key issue to address was perception and education. Most in the digital library community are used to being just consumers of open source software rather than actively involved; other important open source digital library projects, such as FEDORA[3] and the eprints.org software[2], are moving towards a community development model but are at a different point in their lifecycles as open source software projects. Others are concerned about transferring the copyright over work they produce to HP and MIT. Another issue was to do with finding and making the right use of technical infrastructure to support the community. The following sections describe the way we addressed these problems.

2.1 Copyright and Licensing

In addition to a change in mind set, open source presents a new set of legal issues to the digital library community. Typically, library professionals need to be familiar with copyright law as it relates to published printed material. The issues encountered around software licensing and copyright are quite different.

At the present time, the copyright on DSpace is jointly held by Hewlett-Packard and MIT. We currently require contributors to hand over copyright on their contributions to HP and MIT, so that the whole DSpace source code base has the same copyright holder. This has raised concerns and challenged institutions' policies regarding intellectual property. Some institutions are nervous about transferring copyright of work they have done to a commercial company, particularly to be further distributed via a 'commercial-friendly' license like the BSD license. Other institutions have intellectual property policies which either prohibit such transfers or require that they go through some review process. Many are concerned about losing 'credit' or recognition for their work. These are legitimate concerns which should be addressed.

The BSD license was chosen carefully. We wished to allow for vendors to play a role in the repositories area, and by employing the BSD license we hope to encourage the resources of those vendors to work with and benefit the DSpace community, instead of forcing them to create their own, competing systems. Keeping a heavily-modified version of DSpace up-to-date with the evolving core code base can be time-consuming. Contributing modifications back to the core code base so that they are part of that core means that the community as a whole can take over maintenance. This means that both commercial vendors and universities alike will benefit from contributing to the core code base. Thus, the BSD license encourages commercial vendors to become valuable contributors to the core code base, without removing their ability to develop separate components and services with commercial licenses.

We are already seeing the benefit of this; some commerical services are being built around DSpace. HP India offers a commercial service to set up DSpace instances at universities, and BioMed Central's Open Repository service offers set-up and ongoing hosting services. Both of these activities are benefiting the

DSpace community, by increasing the number of users, stakeholders and developers working with the DSpace platform.

The copyright transfer requirement has raised some concern for potential contributors, particularly since not all open source projects have this requirement. For example, the Eclipse project allows contributors to retain copyright for their contributions. There are two principal reasons for requiring copyright transfer: One is to simplify the legal process surrounding copyright or license infringements; a second is to enable the code base to be re-licensed and the copyright transferred in the future.

In general, sizeable open source projects need a legal entity to represent it in any situation where the project's copyright or license is infringed, or is alleged to have infringed some other license or copyright. More copyright holders on the code could be a real hindrance here, as all such holders may potentially have to be involved in such proceedings.

Additionally, as previously discussed [8,7] we are considering long-term stewardship of the DSpace platform and community outside of HP and MIT; however there are is no concrete plan or timeframe at the time of writing. In order to achieve change either the copyright owner of the DSpace code or the license, the consent of all existing copyright owners would be required. If all incoming contributions had separate copyright holders, this would rapidly become a time-consuming logistical challenge. Thus, in order to remain flexible in this regard, fewer copyright holders is better.

Although the fact that HP and MIT are copyright holders may have initially caused concern for potential contributors, a better understanding of the BSD license reduces this. Also, HP and MIT appear to have successfully communicated that they are merely stewards of the code and are not seeking to 'own' it; this, combined with the fact that HP and MIT fulfil the role of the required legal entity to own and license the code, means that the need to set up or join a third-party organisation or foundation for this purpose is not urgent. However, this is still the intention.

2.2 Development Roles

To address the perception that DSpace was still a centrally-developed, free but immutable 'product' from HP and MIT, in April 2004 we introduced a simple development structure around DSpace based on the Apache Foundation model. We defined a group of *committers*, so-called because they have authorisation to *commit* changes to the DSpace source code repository, which included members from outside of HP and MIT Libraries. HP and MIT Libraries both have just one representative in this group, demonstrating that we consider ourselves peers in this community. The group consisted of five members initially, now seven.

Others who contribute to DSpace are *contributors*; this is a role that requires no 'approval' or special status, anyone who helps out is automatically considered a contributor. It is important to note that people can contribute more than just code; experiences, technical support, bug reports, content, and documentation are all needed and welcome.

As we considered and implemented this change in the way DSpace was managed and developed, for a time, the pace of development slowed. This happened when the HP and MIT teams working on DSpace started to focus less on the core DSpace platform, and more on trying to develop the open source community and goals specific to their own research and deployment objectives. Suddenly, it was no one's full time job to look after the DSpace code base. Although neither HP nor MIT disengaged completely from this core maintenance and development process, the geographically dispersed DSpace committer group had to figure out how to progress things and leverage the wider community, while achieving the other objectives of their organisations, which was a significantly different situation from a focussed, co-located team.

However, no one steps up to fill a gap if no gap is there; this slowing may have been necessary to encourage people to step up and contribute. Previously, if someone had made a contribution when HP and MIT had several full-time developers working on the code, they may have seen their contribution lost in the noise somewhat. Now, contributions could be seen to have a real noticable effect; this gives people a rewarding feeling of control. Further, it is noticed by others that HP and MIT are no longer the only developers (in fact HP and MIT developers are now a minority!), and this inspires others to consider contributing.

This change, combined with the development of the infrastructure and perception described in other sections, has resulted in the last year seeing a huge increase in the number of external contributions to DSpace, ranging from simple bug fixes to sizeable feature enhancements. The number of people who have worked on the DSpace code has increased from an initial five developers to around twenty-five at the time of writing, from varying organisations, and is constantly increasing. In addition to numerous bug fixes, some features developed outside of HP and MIT are:

- Customisable submission forms – each collection can have a different set of metadata entry fields
- Image thumbnails can be viewed in item display pages, in search results and while browsing title, date and author indices
- Users can add comments to items in DSpace, a little like the discussions that accompany news items on `slashdot.org`
- Support for LDAP authentication
- Support for internationalised Web user interface

Discussion around many further enhancements continues on the discussion lists, and many groups have indicated that they are engaged in working on those enhancements, so there is no reason to suppose that the number and pace of these enhancements will slow in the near future.

One technical challenge we face is managing those enhancements that involve updating the relational database schema in DSpace. Since applying such changes to an existing DSpace instance requires careful management and taking down that DSpace instance for a time, careful release management is required. To balance this required control with the open development model we have introduced the notion of deadlines for submitting contributions to be included in

a particular release; if a contribution is to be considered for inclusion in the next DSpace version, it must be submitted by a particular date. Then, at that date, we have a fixed set of updates to manage and include in a beta-testing cycle for the next release.

2.3 Infrastructure

It takes more than a Web page and good intentions to support an active open source community. It also needs actively managed collaboration infrastructure which allows a community to form and function cooperatively. It's also important not to have *too many* communication mechanisms available, as they become difficult to keep track of, and the community becomes fragmented. The DSpace community has found the following tools invaluable in this regard.

SourceForge provides many basic functions an open source project requires:

- A publicly accessible CVS repository for managing source code and documentation
- Bug and feature request tracking systems
- A 'patch' tracking system
- Mailing list server and archives

In addition, it is the place where the DSpace software can be downloaded. In general, SourceForge has proved sufficient for all of the above requirements.

Mailing lists are the fundamental means of commication between the geographically diverse community. There should be enough different mailing lists so that not everyone is swamped with traffic not relevant to them, but not so many that the communication is too disparate for a real community to form. It is also important to provide an in-road for those who are interested in the project, but don't necessarily want to sign themselves up to receive dozens of e-mails every day.

For DSpace we have found a good balance with a number of lists:

- A general-purpose, catch-all list for non-technical discussion around the platform and its application and announcements. This list has proved to be fairly low-traffic, which has the benefit that subscribers who are peripherally interested in DSpace are not put off by large volumes of posts. At the time of writing, this list has over 500 subscribers.
- A general technical support and discussion list for those deploying the system and performing local modifications. This tends to be a high-traffic list, and is also the area where the 'community in action' effect can be most felt. It is rare that a request for help on this list goes unanswered, although since support is offered voluntarily by list members as they have time and knowledge to do so, occasionally this does happen. This can leave the poster feeling frustrated or left out; however this is happening less and less, particularly as posters are becoming far more proficient at reporting problems with sufficient information for analysis (as opposed to simply, "I got an error, how do I fix it?") At the time of writing, this list has around 500 subscribers, with over 3,200 messages having been posted over its lifetime.

- A list for developers working on the DSpace platform itself. This is where details around bugs, new features, architectural issues and so forth are discussed. Although there is often overlap with the above technical support list, the discussion does tend to get more involved in this list, so it is useful to have this separate list for those who wish to get that bit more involved with DSpace. At the time of writing, this list has around 120 members and over 1,000 posts.
- A closed list for the DSpace Committer group. This list is primarily used to discuss matters to do with policy and procedures, where members feel a public post is not appropriate. However, in general, discussion happens on the above developer list; as with the source code itself, these processes can benefit from review and input from a wide community.
- Several 'special interest group' lists related to particular areas of application of the DSpace platform. The traffic on these lists is low at the time of writing. Although such lists could potentially serve to fragment the community, in fact they are useful for involving people at the periphery of DSpace, perhaps just interested in one aspect, such as digital preservation. These lists provide a useful level for these individuals to become involved with DSpace, without being exposed to a deluge of low-level or orthogonal technical message traffic.

The DSpace Wiki is proving an invaluable tool. Although the archives of the various e-mail lists above contain a lot of information, in general finding relevant information in them is somewhat awkward and time-consuming due to their volume and unstructured nature. The Wiki has allowed information to be collected and disseminated that is far easier for people to access, and in an expedient and collaborative away not possible with a typical Web site with a small group of maintainers.

In particular it has enabled the creation of some valuable resources:

- A list of projects and people working on DSpace. Anyone can (and is encouraged to) add their work to this list. This means those interested in a particular area of DSpace can find other people working on the same area, encouraging collaboration and minimising duplication of effort. It additionally gives an idea of the amount and breadth of work happening on and around DSpace.
- A list of DSpace instances, to which people can add their own. Seeing a large number of organisations actively using DSpace gives people confidence in the platform.
- Guides and FAQs for developing with the DSpace software. As the processes and practices for this evolve quickly, the Wiki provides a useful place for up-to-the-minute information on this, which people can correct and annotate with experience.

Two features of the Wiki have proved essential—access control, and automatic e-mail notification of updates. Although the Wiki is essentially open for anyone to edit, this was abused for a time as 'spam' was repeatedly posted on the front page of the Wiki. Fortunately, very minimal application of access control (merely securing the front page) has eliminated this.

Every subscriber to the DSpace developer email list receives notification whenever a page on the Wiki is updated. Although this received some initial resistance due to an increase in email traffic, it serves a very useful purpose by alerting people not only to specific changes in the Wiki, but also to the fact that the Wiki is there and being actively used. This last point has been key to the success of the Wiki.

The `dspace.org` Web site is a stable reference point for the project, from which the resources described above can be reached. It also provides some background material about the project, and a considerable amount of guidance from the experience of MIT for universities on the non-technical aspects of creating an institutional repository service using DSpace. This site gets about 25,000 visits a month.

These non-technical implementation aspects, while important, appear a far less active and dynamic area of discussion on the DSpace mailing lists. This is probably because things like assembling resources and defining policies take a lot longer to change than source code!

3 Moving Forward: New Challenges and Opportunities

The developer community around DSpace is now starting to function smoothly. We have seen a considerable increase in technical activity around DSpace recently. New patches (code contributions) are received every week. In addition to these feature enhancements and bug fixes that arise from a particular organisational need, various significant research projects around DSpace are underway.

The DSpace/SRB integration project [5] at the San Diego Supercomputer Center and MIT is investigating using Grid storage technologies with DSpace. Specifically, they are using SDSC's Storage Resource Broker technology, although the intention is to make storage 'pluggable' so that a variety of storage mechanisms can be used, from simple file systems to large-scale distributed storage.

SIMILE is a joint MIT and W3C project [6] looking at employing Semantic Web and RDF technologies to support heterogenous metadata storage and indexing in DSpace.

DSpace@Cambridge is looking at various long-term digital preservation issues, such as managing file formats and auditing processes, and also working on support in DSpace for e-learning activities.

CWSpace [1] project to investigate archiving OpenCourseWare course materials in DSpace, as well as the related standards and interoperability protocols to support reuse of those materials in course management systems and collaborative learning environments.

The University of Minho [4] in Portugal are experimenting and developing several add-ons to DSpace, including ontology support for classifying items and tools to visualise relationships between content and researchers.

In addition to projects such as those listed above which explicitly involve DSpace as part of the deliverables, many other projects have adopted DSpace as a platform on which to base prototypes and research, such as the EU-funded Digital Academic Repositories (DARE) project in the Netherlands and the Australian Partnership for Sustainable Repositories (APSR) project of which the Australian National University is the lead insitution.

Open source provides these research projects with an exciting new opportunity to see their results directly reflected in a functioning system that is deployed at over a hundred organisations. Their work will be visible and useful to the thousands of end users of these systems.

Along with this new opportunity comes a new challenge: coordinating the results of the various projects. There is a need for these projects to collaborate so that effort is not unduly duplicated, and more importantly, they move the platform as a whole in a consistent and beneficial direction. This is driving some work looking at improving the underlying architecture of the system.

3.1 Architectural Evolution: DSpace 2.0

One of the reasons underlying DSpace's widespread adoption is that it is an end-to-end application. Every basic function required to deploy and operate an institution repository or similar system is present, though not necessarily to a sophisticated degree. This approach was taken because it was not known *a priori* exactly where 'development dollars' would best be spent; such knowledge comes from the experience of building, deploying and operating the system. This, coupled with the unanticipated extent of adoption of DSpace, has resulted in the need for some evolution of the underlying system architecture of DSpace in addition to the development of added functionality, particularly in two areas.

Storage — In the current architecture, all metadata is in a relational database, and all content bitstreams are in the file system on the server. This makes certain preservation-related activities complex, including:

- Backups — coordinating the backups of bitstream storage and relational database. A backup must contain a snapshot of the relational database and the bitstream store in a consistent state; i.e. the contents of the bitstream store backup must be as the data in the relational database expects. To make a reliable back up, one needs to take a snapshot of both the database and the bitstream store at one moment in time, and this involves freezing both for the duration of the backup process.
- Auditing — Although it is simple to audit the bitstreams in the bitstream store, auditing the metadata and the structure, for example checking that all items are present and correct, is not.
- Replication/distribution — Any replication or 'sharing' of content, metadata etc. will require processing at every stage to extract metadata from the relational database and package it with the appropriate content.

Modularity — The diverse community using DSpace needs to be able to deploy different flavours of DSpace, for example one that uses a particular

identifier scheme; researchers and developers need to be able to develop and experiment with DSpace without unduly affecting others or being overly encumbered with the maintenance effort involved in keeping their customisations up to date with the core DSpace code base.

To address these requirements, an evolved DSpace architecture dubbed 'DSpace 2.0' was presented at the first DSpace user group meeting held in Cambridge, MA, USA in March 2004 [9]. This updated architecture proposed three major areas of refactoring the DSpace architecture:

Asset Store — a component which stores Archival Information Packages, consisting of the serialised metadata and bitstreams in a DSpace item. This means objects in DSpace are more self-contained in the system, enabling easier backup, auditing and replication. The metadata in these AIPs may be replicated in a relational database performance reasons, however the AIPs are the 'authoritative' version of the metadata and content, with the relational database copy being considered a cache.

Module Layer — The various components of the DSpace system are more cleanly modularised; they interact only via defined APIs, and an individual instance can plug in whatever implementation of each API they like.

Web user interface — This needs to be modularised, as most customised functionality must be reflected in the user interface. The Apache Cocoon publishing framework was proposed as a potential candidate for addressing this, as Servlet/JSP has proven to be wanting in this particular area.

This design proposal was largely welcomed by the open source community; however, it wasn't a full specification. Immediately returning to a 'closed shop' style of development would cancel out the benefits of having developed an open source community around DSpace, so we are presented by a number of challenges:

- Establishing consensus around technical issues when there is no single 'technical lead' or decision maker.
- Finding resources to do the (considerable) required development work.
- A large existing user base has invested in understanding and modifying the existing DSpace system. Introducing a completely new version of the system could alienate some of these people; at best, it would divide the resources of the DSpace community between essentially two different systems.

For this architectural work, a new development model has started to emerge, based around prototyping. Developers at institutions with a vested interest in a particular aspect of this refactoring produce a small prototype demonstrating a particular approach to one aspect of the architecture.

This breaks down the refactoring task into managable pieces. Building and sharing these prototypes brings to light the various underlying technical issues, many of which are not apparent during prior design work and discussion. Building such prototypes also represents a relatively small developer commitment;

they certainly do not require the level of consensus, coordination and resources that constructing a complete system does.

Various prototypes have been built thus far:

- Various prototypes for the DSpace 2.0 asset store, built by individuals at HP, MIT and Cambridge University
- A prototype user interface based on Apache Cocoon, built at Australia Nation University
- A prototype ingestion system based on Apache Cocoon, also built at Australia Nation University

These prototypes may mature and be folded into the DSpace code base directly, or they may simply serve as 'proofs of concept' serving as input to further development efforts. Where more than one prototype represents a possible route forward, the task of achieving consensus should be facilitated by the availability of functioning code.

This prototyping approach provides a path forward in achieving the DSpace 2.0 architecture; it does not address all of the challenges mentioned above. The problem of how to introduce these architectural changes with a large existing user base still remains. We may need to pursue a gradual, evolutionary approach; it seems unlikely that the current community would be able to support two separate versions of the system.

4 Conclusion

With DSpace we have made considerable progress in moving from a typical, closed team style of development to a wider, open source model where the responsibilities for maintenance and development are held by the community as a whole. This brave new world has presented us with many challenges, and it has not been an easy path; for a time, the pace of development of the DSpace platform slowed considerably. However, this has been a necessary phase. It takes time to establish processes and infrastructure to support open source development; slowing development is also a factor in encouraging members of the community who are used to being only consumers to become active contributors to the effort.

We have made considerable progress in addressing this: The fact that DSpace contributors from the wider community outnumber those inside HP and MIT is a testament to the fact that we are functioning as an open source community. DSpace now has a key attribute of open source projects: *vitality*. These days, open source projects are judged more in terms of the activity around them than the merits of the technology itself; a system which still needs work but has a bustling community around it is likely to be a better long-term bet than a more technically developed system with no visibly active community.

While we have overcome some of the hurdles we reported last year, we face some new challenges. A potential emerging challenge is around research projects looking at significant architectural directions and decisions for DSpace. How do we manage the case when consensus of the community as a whole is to move in a

different, perhaps contradictory direction? How do we still engage the resources of that project, which is obliged to explore its own direction? Will such projects regard the fact that their proposed directions were not adopted as 'failure'?

An additional challenge exists around deeper architectural development: Since DSpace version 1.0 was built as a 'breadth-first' system, it was not perfect in every regard; some work on the underlying architecture to facilitate preservation and modular development are needed. An evolved DSpace architecture to address these requirements has been proposed and accepted; however, making large-scale changes in the system in an open source environment with a large number of existing users is proving a challenge. We are making considerable progress in this regard, as the open source community breaks down the problem and constructs small-scale prototypes of various aspects. This provides us with a grounding for discussion of the technical issues, and a concrete basis around which to build consensus.

As well as new challenges, it is also becoming clear that the open source model of development around DSpace is providing researchers with an exciting new way to see the fruition of their labour. Although publications, presentations and prototype systems are useful, too many good ideas come to a halt at this stage. The DSpace open source platform provides a means for these researchers to see their work deployed and used by a wide audience in a way that was difficult or impossible for this community before.

Although some challenges remain, the open source model of development brings numerous new and exciting opportunities to the digital library community.

References

1. CWSpace. http://cwspace.mit.edu/.
2. The eprints.org software. http://www.eprints.org/.
3. The flexible extensible digital object and repository architecture project (FEDORA). http://www.fedora.info/.
4. DSpace Dev @ University of Minho. http://dspace-dev.dsi.uminho.pt:8080/en/welcome.jsp.
5. Reagan Moore, Richard Marciano, MacKenzie Smith, and Brian E. C. Schottlaender. NARA supplement to the NPACI collaboration: Integrating data management with data grids. http://libnet.ucsd.edu/nara/, September 2004.
6. SIMILE: Semantic interoperability of metadata and information in unLike environments. http://simile.mit.edu/.
7. MacKenzie Smith. DSpace user group meeting summary and outcomes. http://www.dspace.org/conference/meetingsummary.html, March 2004.
8. MacKenzie Smith, Richard Rodgers, Julie Walker, and Robert Tansley. DSpace: A year in the life of an open source digital repository system. In *Proc. 8th ECDL*, pages 38–44, Bath, UK, September 2004.
9. Robert Tansley. DSpace 2.x architecture roadmap. http://www.dspace.org/conference/presentations/architecture.ppt.