Infrastructure for Interactivity - Decoupled Systems on the Loose

Andreas Aschenbrenner[1], Flavia Donno[2] and Senka Drobac[3]

[1]Andreas Aschenbrenner, Goettingen University, Germany, e-mail: aaschen@gwdg.de
[2] Flavia Donno, Grid Support Group (IT), CERN, Geneva, Switzerland, e-mail: Flavia.Donno@cern.ch
[3] Senka Drobac, Ruder Boskovic Institute, Zagreb, Croatia, e-mail: senka.drobac@irb.hr

Abstract—**Digital ecosystems are not "created", they form and evolve wherever their users guide them. Moreover, usage patterns within digital ecosystems are not bound to one particular technology. A computational neurologist, for instance, may move back and forth between laboratory experiments, grid infrastructure for simulation and analysis, collaborative environments and publication platforms.**

This paper pinpoints an evolution of various ecosystems that is currently ongoing, and discusses technological patterns for mixing and merging infrastructures. In particular it looks at "repositories" as they overarch scientific infrastructure and interactive applications. In an analysis covering a series of experiments, we find an optimal setup in the combination of grid and web technologies through a REST-based interface, which opens up a variety of novel architectural patterns.

Index Terms— **repository, grid, cloud, pattern.**

I. Introduction

This paper addresses the area between two well-established technologies that brought a variety of digital ecosystems into being. Digital infrastructure in scientific contexts is often associated with virtualizing hardware resources through grid technologies. Grids have been working towards maximum performance and automation to tackle hardware-intense computing challenges of simulations, major experiments, and the like. However, scientific infrastructure is increasingly expanding to also accommodate interactive services. Scientific workflows and interactive visualizations are just the first step on this path.

Web environments on the other hand have been all about interactivity, user-generated information and references. To achieve this, web technologies have been defined by simplicity to empower non-expert users. Mash-ups and clouds emerge as part of web environments, which accommodate increasingly resource-demanding interactive services on the web.

In the following we enter the space between these two contexts and usage patterns - infrastructure for large-scale scientific applications on the one hand, and open environments for interactivity and user-generated content and services on the other. We are not pioneers in this space; we are merely followers of what is out there already: high-volume video platforms for e-learning, music portals that analyze audio patterns for listening recommendations, and many of the other services that populate our digital world 2.0. However, while the two usage patterns are converging (high-volume and automation vs. simple and interactive), the technologies are slow in taking up experiences from each other. We therefore ask the question: As the two paradigms are starting to converge towards a common view of the world, how do their technologies interact? How do you mix grid resources into the world wide pond of mash-ups?

II. Repository-Based Ecosystems

In our analysis we take "repository" technologies as a case study and look at possible adoption patterns between the two worlds. (a) Data repositories in scientific contexts such as the often-cited large hadron collider experiments (LHC) [1] are high-volume stores for primary data implemented on grid resources and embedded in automated workflows. (b) Repositories for open access publications are, on the other hand, web portals that guide interactive deposit processes by authors and consequently preserve a multitude of digital objects consisting of document-size files. While following similar terminology, the two respective world views are unlike each other.

Despite their different origins, both technologies are converging: (a) Data stored in scientific infrastructure is being unlocked for open access and interactive applications, and (b) publication repositories are poised to accommodate research data and workflows as well.

TextGrid[1] embeds interactive functionalities into a Globus-based grid environment[2]. Storage is handled by existing Globus data grid functionalities. Upon this storage infrastructure, mechanisms for storing and retrieving metadata, object behaviors, interactive workflows, and the like were modeled after open access repository systems like Fedora[3]. While building on both, grid technologies and repository concepts yielded a solid, multi-purpose platform, TextGrid is looking for further technology convergence between the two contexts.

DARIAH[4] establishes a digital research infrastructure for the humanities in Europe. It is composed of a number of humanities data archives that aim to federate their holdings and jointly build a distributed virtual repository. Massive amounts of data expected from image and video assets are best stored and replicated in national grid infrastructures.

[1] TextGrid. www.textgrid.info
[2] Globus. http://www.globus.org/
[3] Fedora - Flexible Extensible Digital Object and Repository Architecture. http://fedora-commons.org/
[4] DARIAH - Digital Research Infrastructure for the Arts and Humanities. http://www.dariah.eu/

Thus, while DARIAH builds primarily on Fedora and its content modeling capabilities, it seeks integration with grid-based storage infrastructure.

These projects - as well as others [2] including Shaman[5], eSciDoc[6], and D4Science[7] - exhibit the following traits and more:

(1) work with large volumes of research data that is to be backed up and replicated across distributed locations in order to ensure bit-preservation.

(2) require administrative workflows to ensure proper ingest and indexing of the data as well as long-term maintenance of the archive.

(3) are open to scientific workflows as well as (external) interactive applications, possibly as entry-points into a variety of different virtual environments.

(4) may federate with other repositories and services to share content through open interfaces such as those provided by the Open Archives Initiative (OAI)[8].

In our search for suitable patterns that combine infrastructure and repository technologies, we probed various approaches: using Cleversafe[9] as a virtual file system, iRODS[10] integration, as well as a RESTful abstraction upon the Storage Resource Manager SRM[11]. While we cannot provide a silver bullet to all questions involved, this analysis may help projects confronting similar questions by offering patterns for integrating scientific infrastructure with interactive applications.

A. Cleversafe, transparent storage

Cleversafe dubs itself a "dispersed storage" network, which was available in version 1.1 at the time of writing (April 2009). Cleversafe is open source software developed by a company as their key product. At the basis of the Cleversafe software is an algorithm, which chunks data into pieces, spreads them over data nodes, and performs error correction for fault tolerance (a Reed-Solomon Code). This algorithm displays stability in the face of failing nodes, good read performance from redundant, distributed nodes, and increased security as single nodes only merely accommodate encrypted data chunks. A Cleversafe storage 'vault' can be mounted via iSCSI and hence works as a virtual file system.

We used an early version of Cleversafe (Version 0.7.8, November 2007) and installed six virtual CentOS 5 nodes for the storage network. This purely experimental setup, in addition to the fact that we used an early version of the software resulted in rather slow access rates - hence, performance is no criterion in this experiment. Cleversafe was used to store the digital objects of a Fedora 2.2 repository, running on Ubuntu Linux. No adaptation of the Fedora repository was necessary for this, the Fedora installation was out of the box.

Regarding the technical requirements formulated above, Cleversafe promises to scale to any size and offer support for storage management, and it can be distributed. A single Cleversafe network can only be shared read-only, since multiple iSCSI initiators writing data at the same time could compromise data consistency. Hence, a journaling configuration with fail-over can be installed easily, yet multiple entry points for ingest cannot be provided on an infrastructure level.

Cleversafe offers general purpose, replicated storage, yet it does not facilitate repository-specific functionality through administrative workflows or abstraction of repository data management (folder structure, file naming conventions, etc.). Moreover, due to the distribution algorithm, it is hardly possible for adapting Cleversafe accordingly.

B. Opening up iRODS

Where Cleversafe offers an austere data grid with transparent replication, iRODS is functionality-wise clearly at the other end of the spectrum. Developed by the DICE team[12] around Reagan Moore - formerly at the San Diego Supercomputing Center, now at University of North Carolina, Chapel Hill -, iRODS is a data grid software system. So-called rules that are capable of triggering microservices allow comprehensive adaptation of administrative workflows and hence of tailoring the data grid to the respective application environment. Besides low-level data grid and administration functionality, it intends to offer graphical applications such as an AJAX-based web interface as well. As such it offers the whole stack of repository functionality, from low-level data management to user interfaces.

Despite this broad spectrum of activities, iRODS has not yet comprehensively addressed interactive functionalities, particularly workflows such as ingest procedures for authors or more sophisticated object modeling capabilities. For example, the DSpace[13] repository offers a comprehensive user community model, and Fedora offers more advanced metadata and content modeling mechanisms. Various projects are hence looking into combining iRODS with DSpace respectively Fedora.

Various ways on how to integrate iRODS with repositories such as DSpace and Fedora are conceivable. The following integration scenarios refer to the four requirements cited in the introduction to this article:

1. **iRODS objects as external datastreams** - some repositories including Fedora are capable of managing the metadata of digital objects that are outside of their stores. While this allows for referencing objects stored in an iRODS data grid and possibly attaching behaviors to them (e.g. Fedora [3], aDORe DIM [4]), the

[5] SHAMAN - Sustaining Heritage Access through Multivalent ArchiviNg. http://shaman-ip.eu/

[6] eSciDoc. http://www.escidoc.org/

[7] D4Science - DIstributed colLaboratories Infrastructure on Grid ENabled Technology. http://www.d4science.eu/

[8] OAI - Open Archives Initiative. http://www.openarchives.org/

[9] Cleversafe. http://www.cleversafe.org/

[10] iRODS - Integrated Rule-Oriented Data System. https://www.irods.org/

[11] SRM - Storage Resource Manager. http://sdm.lbl.gov/srm-wg/

[12] Data Intensive Cyber Environments Research, DICE. diceresearch.org

[13] DSpace. http://www.dspace.org/

repository has no means for managing the object (audit trails, versioning, etc). - *requirement 1 (iRODS); 2, 3, 4 (repository)*

2. **using iRODS as a repository storage module** - instead of storing data locally on the the repository server, iRODS provides a distributed storage layer. The Jargon Java API for iRODS is used for directly implementing the storage handler into the repository. DSpace offers support for the Storage Resource Broker[14] and iRODS as part of its general release. Furthermore, in a joint effort the DSpace and Fedora teams aim to develop a generic storage handler with a plug-in mechanism [5] to accommodate iRODS and just any other storage handler. However, while this possible integration is acknowledged, it falls short in using available capabilities in iRODS for rule management. - *requirement 1 (iRODS); 2, 3, 4 (repository)*
3. **iRODS microservice and rule support** - one step further from a simple virtual storage, iRODS could define rules for parsing a newly deposited object on ingest, extract the metadata into the ICAT database, and hence activate its low level rule support. With all the features available in iRODS, this is technology-wise a minor step to take, yet demands a higher level of coordination between iRODS and the repository: metadata management is being replicated in both, iRODS and the repository, and they need to be synchronized. Behaviors on iRODS and the repository level must not interfere with each other. - *requirement 1,2 (iRODS); 2, 3, 4 (repository)*
4. **integrating iRODS into the repository landscape** - since iRODS is offering the complete stack of repository functionality up to user interfaces, iRODS could fully integrate into the emerging repository landscape. Standards such as Zing[15], OAI-PMH[16], and OAI-ORE[17] have shaped and will continue to shape an ecosystem of federated repositories with meta-portals and external service providers. However, iRODS is not currently offering any standards-based API's. - *requirement 1,2,3,4 (iRODS); 2, 3, 4 (repository)*

While there may be other approaches, these four patterns exemplify various integration levels of infrastructure and repositories - integration of specific systems as well as integration in open ecosystems with a variety of technologies and interests. Please note the dramatic difference between the first and the last pattern with regard to openness. While the first pattern is defined by the openness of the repository, the last one offers standards-based entry points on an infrastructure level, thus fostering the emergence of attached repository environments.

It is obvious from the above that infrastructure like iRODS and repositories like DSpace and Fedora are starting to overlap functionality-wise. Despite a possible loss of synergies, overlapping technologies are no problem as such. On the contrary, if usage patterns and communities overlap as well, this may be a breeding ground for standards, mutual services and other components effective ecosystems build upon.

C. SRM in an S3-like abstraction

Instead of seeking an even larger and even more comprehensive system, this approach returns to the very core of scientific infrastructure. The Storage Resource Manager (SRM)[18] [6] stems from the high energy physics community where it serves as one of the grid middleware components to distribute the massive data influx from experiments such as the Large Hadron Collider at CERN[19]. The SRM is a standard protocol for initiating transfer between storage resources. It therefore is capable of mediating between storage components of various types, transfer protocols, and other grid components. To sustain the large amounts of data from experiments, SRM is geared towards performance and works on a block level, providing low-level functionality on files and storage space. Specialized functionalities include pinning of files (i.e. locking for a defined time), space reservation, and others.

SRM is a versatile, pivotal grid standard, and it is highly interlinked with its environment. We first evaluated whether SRM could be plugged into a repository as a storage handler, similar to how iRODS can be plugged into DSpace. Since SRM is not a transfer protocol itself but a mediator, transfer protocols such as GridFTP, DCache dccp, or globus-url-copy are required as well. When operating in a grid environment, the user needs a grid certificate as well as authentication mechanisms such as grid-proxy-init and the like to authenticate. Hence the certificate as well as security related protocols have to be available at the client. The officially supported client library called GFAL[20] - Grid File Access Library - is based on C and Python[21], respectively the package "lcg_util" provides convenient command-line tools.

As an interface between a grid node (SRM) and a web server (the repository), we hence looked for a lightweight interface that is capable of translating between the two worlds. Usage patterns for repository-based applications clearly differ from those needed in scientific infrastructure. Performance requirements are absolutely central in the latter, whereas in web contexts we can compromise on some of the tuning parameters in order to simplify communications. Existing cloud services are a premier model for translating between infrastructure and the web. Cloud services like those by Amazon[22] offer both, REST and SOAP-based interfaces. Despite its simplicity, REST-

[14] SRB - Storage Resource Broker. http://www.sdsc.edu/srb/index.php

[15] Zing, SRU/SRW. Library of Congress. http://www.loc.gov/standards/sru/

[16] OAI-PMH, Open Archives Initiative: Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/openarchivesprotocol.html

[17] OAI-ORE, Open Archives Initiative Protocol: Object Exchange and Reuse. http://www.openarchives.org/ore/

[18] SRM - Storage Resource Manager. http://sdm.lbl.gov/srm-wg/

[19] CERN openlab for DataGrid applications. www.cern.ch/openlab

[20] Grid File Access Library - GFAL. http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/data_gfal.html

[21] respectively other programming languages through SWIG (Simplified Wrapper and Interface Generator), http://www.swig.org/

[22] Amazon Web Services. http://aws.amazon.com/

based protocols satisfy all the needs of the web community.

Instead of defining yet another API, we hence decided to re-engineer the REST API of the Amazon S3 storage service as an interface between the grid environment and the repository. An experimental implementation of the S3 interface uses Python WSGI (Web Server Gateway Interface)[23] and works fine - S3 libraries like Jets3t[24] can be re-rooted from the Amazon cloud to our re-engineered cloud-like interface. There already is a DSpace storage handler implemented upon the Jets3t library,[25] which is planned to be implemented upon our re-engineered cloud. First, however, we will take performance measurements and analyze the feasibility of this setup in a productive environment. Further analysis will be made available in the next few weeks. While also not part of the experimental setup, authentication, of course, is an important issue. We are observing closely the progress of projects like IVOM[26], which aim to integrate Shibboleth[27] into grid environments. Shibboleth allows for single sign on of users via their home institution. The approach for Short Lived Credentials in IVOM and similar projects appears to be very promising also for an architecture as sketched here. The S3 protocol also contains the two configuration options "location" and "storage_class". While the latter is apparently unused by Amazon, it offers a mechanism to define replication policies on a repository level which can then be executed in the infrastructure accordingly. One possible scenario may be a storage class "confidential, valuable", which triggers the infrastructure to replicate the asset in three distributed data centers with particularly high security measures, rather than the two copies made in the case of a "standard" storage class.

The advantages of such a loosely-coupled, HTTP/REST-based architecture are manifold:

The interface between the repository and the cloud-like service is obviously very light-weight. Due to the loosely-coupled architectural paradigm, the interdependencies between infrastructure and application (in this case: the repository) are minimized and the two can evolve separately.

So why not build on Amazon S3 from the outset? Using S3 is an option, of course, yet most of the research data we are holding is unique and valuable. While Amazon promises multiple copies of each file and an uptime of more than 99%, we don't know whether Amazon will still be there and offer on-demand storage in 20 years. Even replicating to and switching between multiple cloud providers is no option to us. We have substantial storage resources locally and within the D-Grid national computing infrastructure. Moreover, we appreciate some control over the infrastructure both organizationally and technologically, which allows us to advance the interface to include specialized functionalities as well.

A generic repository storage API and a decoupled architecture pattern like this enables other services to tie into the system environment. Multiple repositories can build on a single storage, and even specialized services e.g. for format conversion or other administrative tasks are conceivable to work directly at the level of the S3 API. Administrative workflows triggered by the repository, yet executed on the storage level may boost overall scalability of the system environment considerably. Moreover, this loosely-coupled approach may trigger the creation of low-level repository services and hence a variety of agents interacting in an open repository ecosystem.

III. LOOSELY-COUPLED PATTERNS

The three experiments outlined in the last chapter moved from a purely localized system, to a tightly integrated system with a proprietary API, to a light-weight and open service architecture. This catharsis resonates with similar experiences in other contexts. Initiatives looking at the big picture such as the JISC Information Architecture (2005, [7]) noted that repositories are part of a larger environment of services, including authentication/authorization services, format and service registries, and various other components. Even the components defined in the JISC architecture are expected to be supplemented with other components over time. Repositories are by nature open environments, as opposed to controlled systems with defined borders.

An open interface such as the S3-derivate described above is, of course, not yet an open environment of loosely-coupled services. So what kinds of architectural patterns are conceivable with this generic, HTTP/REST-based interface? This section describes two patterns building on the storage interface, which may spawn a variety of different agents and services.

A. Search and analysis

Search is an extremely important functionality for repositories, as it is often the primary entry point for users into the repository collections. The TextGrid project [8], for example, offers a set of search mechanisms with increasing sophistication: keyword search in just all the content; metadata search for specifying authors, title, or other structured information; as well as XQuery-based search[28], which works on the XML data model underlying a given document. All of them are directly at the core of the TextGrid functionality, yet only separation of the search mechanism from the repository core yielded an architecture that was sufficiently scalable for the number of digital objects as well as concurrent users to be expected.

Digital objects ingested into the repository are stored and preserved in a storage vault, based on the Globus Toolkit. Each object is assigned a unique identifier (URI) for retrieval. At the same time, an object's metadata is stored

[23] WSGI - Python Web Server Gateway Interface, PEP 333, v1.0. http://www.python.org/dev/peps/pep-0333/

[24] JetS3t - Java toolkit for Amazon S3. https://jets3t.dev.java.net/

[25] DSpace 2.0/Pluggable Storage. DSpace Wiki. http://wiki.dspace.org/index.php/DSpace_2.0/Pluggable_Storage

[26] IVOM - Interoperability und Integration of VO-Management Technologies in D-Grid. http://www.d-grid.de/index.php?id=314&L=1

[27] Shibboleth - authorization/authentication in web environments. http://shibboleth.internet2.edu/

[28] XQuery - an XML Query Language. http://www.w3.org/TR/xquery/

into a metadata database. If an object is in XML format, it is additionally stored into an XML-database, which allows for the XQuery-based search mechanism. So, the object is stored twice (and the metadata three times), once for preservation and once for analysis, separate from each other. Consistency between the redundant objects is maintained since the storage module pushes all incoming XML-objects directly to the databases.

This approach of redundant storage in separate services may appear obvious in the days of Google. Or also for the experiments in high energy physics mentioned above, which use separate grid nodes for storage and computational analysis respectively. However, it is only possible with an open interface directly on the storage level. Various types of added-value services using this pattern are conceivable, including text mining, data clustering, and monitoring.

B. Preservation services

Another core repository responsibility is preservation. Many of the tasks involved in preservation cannot feasibly be realized by one repository alone. Subsequently, there are numerous preservation efforts shared by the repository community, including the PRONOM format registry[29], the validation and metadata extraction tool JHOVE[30], or mass conversion in the CRiB project[31]. In various initiatives these tools are nested into light-weight preservation services to be attached to repositories.

The preservation services mentioned above exhibit a variety of different patterns. Many of those services are outside of the repository, yet are embedded in repository workflows. For example the metadata extraction for format identification, which is performed on ingest with the resulting metadata being passed along with the object to storage. However, for mass-conversion objects are ideally taken directly out of the storage, processed at a dedicated server, and subsequently returned back for storage. While the action is triggered through the preservation manager from within the repository, the conversion service is outside of the repository and communication is for the most part between the conversion service and the storage environment directly. Again, this harvesting pattern is only possible through open interfaces on a storage level and some level of decentralization.

C. Generic patterns

Looking at the two cases above, both display the traits of usual push and pull patterns. In the search/analysis example, the storage infrastructure pushes immediately a newly ingested object to the external agent. The conversion service harvests data from the storage infrastructure, thus pulling the valuable content out of storage.

In web environments, a pull-architecture is very common in web environments. Roy Fielding, in his dissertation about networked-based software architectures, even states that "the scale of the Web makes an unregulated push model infeasible" [9]. The reason for this is not because one-to-many communications in web environments are generally infeasible, but rather since in an uncontrolled environment the server cannot just push out content to the world since the largest part of the world is just not interested. Push models that work in a web environment is event notification by subscription. Thus, all interested agents could subscribe at the storage infrastructure e.g. for the event "creation of new object", or another CRUD operation (Create, Read, Update, Delete). The subscription pattern is similar to the XML-database example mentioned above, and makes push in a web environment feasible.

Faithful to its strong roots in the web environment, Amazon S3 enables pull, yet no push. S3 offers no event notification in the sense of subscribing to CRUD operations outlined above. To benefit from push patterns, subscription has to be done either on an application respectively a repository level, or an additional notification service is retrofitted onto S3. While avoiding complex functionality at the cost of efficiency or generality, this could be done as an index of all content in the storage infrastructure. This index would contain - apart from the name of the object and maybe few other low-level metadata - also the date of creation and last update. It is the combination of a light-weight, loosely-coupled interface with mechanisms to allow for pull and maybe also push-based patterns, which offers a fertile breeding ground for independent agents and, hence, for a healthy ecosystem.

IV. Related Work

Many fields have been infected by the cloud buzz and are looking for ways to cloud-enable their applications. As this paper outlines, using RESTful infrastructure services in a repository ecosystem is not just a cool fad, but a necessary next step in repository evolution [10]. A RESTful approach clearly excels over the tighly-coupled approaches analyzed in Cleversafe and iRODS.

While some repository systems have been looking into cloud technologies [5, 11], a serious attempt in linking repositories into national (grid) infrastructure has been missing. Similarly, the opportunities for repositories in loosely-coupled patterns have not been explored until recently.

The grid community has been discussing the rise of clouds intensely. When it comes to linking grids and clouds, some grid experts have suggested using clouds as hosts for grid software [12, 13]. The idea for using cloud-like interfaces to provide access to grid resources - as suggested in this paper - has also been picked up, and has been submitted for discussion at the Open Grid Forum.[32]

[29] PRONOM. http://www.nationalarchives.gov.uk/pronom/

[30] JHOVE - JSTOR/Harvard Object Validation Environment. http://hul.harvard.edu/jhove/

[31] CRiB - Conversion and Recommendation of Digital Object Formats, http://crib.dsi.uminho.pt/

[32] Ignacio M. Llorente, Thijs Metsch: Cloud Interface API - BoF. (March 2009) http://www.ogf.org/OGF25/materials/1567/Presentation.pdf

V. Conclusions

Requirements and usage patterns in scientific infrastructure and interactive (web) applications increasingly share common requirements. The approaches for bridging the gap between those two contexts presented here show the way towards decoupling system components.

The Cleversafe experiment was insufficient as to the requirements initially posed, and is clearly surpassed by an iRODS approach. iRODS offers a multitude of functionalities and is designed to support all conceivable requirements for data-intensive grids. The third experiment went another route. Rather than exploring systems combining even more functionality it created an open, generic interface for just anybody to plug into. While providing only little functionality in itself, the open and simple interface potentially triggers the collaboration of various agents. It has the potential of satisfying all requirements formulated at the beginning of this paper, and possibly even more.

As the "Common Repository Interface Group" CRIG[33] persuasively cites: "The coolest thing to do with your data will be thought of by someone else." Openness and simplicity facilitate the interaction of independent agents and may hence be the very core of digital ecosystems. Digital ecosystems of course emerge and evolve over time, but the repository community has the potential of becoming an open ecosystem that overcomes community borders.

Lastly, there has been much discussion about grids versus clouds. This paper shows that grids and clouds may interact smoothly. After all, they both follow similar goals in virtualizing resources and simplifying the lives of their users. While they may address different usage patterns[34] and thus offer different user interfaces, they share similar technical characteristics. From this it seems that merging grids, clouds, and interactive repositories seems like a small step away, and the light-weight interfaces on multiple layers may trigger entirely new patterns and actors in the ongoing evolution of those digital ecosystems.

VI. References

[1] Lana Abadie, et al., "Storage Resource Managers: Recent International Experience on Requirements and Multiple Co-Operating Implementations," In: 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007), 2007, pp.47-59.

[2] Andreas Aschenbrenner, Tobias Blanke, Neil P Chue Hong, Nicholas Ferguson, Mark Hedges, "A Workshop Series for Grid/Repository Integration", D-Lib Magazine, January/February 2009, Volume 15 Number 1/2.

[3] "Content Model Architecture," Fedora Commons Wiki. http://fedora-commons.org/confluence/display/FCR30/Content+Model+Architecture

[4] Jeroen Bekaert, et al., "Using MPEG-21 DIP and NISO OpenURL for the Dynamic Dissemination of Complex Digital Objects in the Los Alamos National Laboratory Digital Library," In: D-Lib Magazine, February 2004, Volume 10, Number 2.

[5] "DSpace Foundation and Fedora Commons Receive Grant from the Mellon Foundation for DuraSpace," HatCheck Newsletter, November 11th 2008. http://expertvoices.nsdl.org/hatcheck/2008/11/11/dspace-foundation-and-fedora-commons-receive-grant-from-the-mellon-foundation-for-duraspace/

[6] Alex Sim, Arie Shoshani (eds.), "The Storage Resource Manager: Interface Specification". Version 2.2, 24 May 2008. http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html

[7] Andy Powell, "A 'service oriented' view of the JISC Information Environment", November 2005, Bath. http://www.ukoln.ac.uk/distributed-systems/jisc-ie/arch/soa/jisc-ie-soa.pdf

[8] "TextGrid: A Community Grid for the Humanities". In: German Grid Initiative, Heike Neuroth, Martina Kerzel, Wolfgang Gentzsch (eds.), Universitätsverlag Göttingen: 2007, pp. 62-64.

[9] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures, Chapter 5". Dissertation, University of California, Irvine: 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[10] Andreas Aschenbrenner, Tobias Blanke, David Flanders, Mark Hedges, Ben O'Steen, "The Future of Repositories? Patterns for (Cross-)Repository Architectures", D-Lib Magazine, November/December 2008, Volume 14 Number 11/12.

[11] David Flanders, et al., "Fedorazon Final Report to JISC". http://www.ukoln.ac.uk/repositories/digirep/index/Fedorazon_Project_Reports

[12] "Grids and Clouds - Evolution or Revolution? An EGEE Comparative Study." EGEE Report, Mai 2008. https://edms.cern.ch/document/925013/

[13] Sergio Andreozzi, Luca Magnoni, Riccardo Zappi: Towards the Integration of StoRM on Amazon Simple Storage Service (S3). In: Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP'07). Journal of Physics: Conference Series 119 (2008). IOP Publishing, doi:10.1088/1742-6596/119/6/062011.

[33] CRIG - Common Repository Interface Group. http://www.ukoln.ac.uk/repositories/digirep/index/CRIG

[34] (homogeneous data in high-performance infrastructure, versus heterogeneous data through simple, general-purpose interfaces for interactive applications)