# A Technology Analysis of Repositories and Services

Sayeed Choudhury
Jim Martino

*This document represents a presentation given at the Spring 2005 CNI Task Force Meeting in Washington, DC on April 5, 2005.*

*Abstract:*

The concept of the institutional repository has gained traction within the digital library community. While this idea provides a useful description that may facilitate institutional adoption, it may also oversimplify the complete picture associated with digital library architecture. Institutions may now be finding that there will be multiple repositories and applications in the same environment. Developing individual interfaces for each application/repository pair presents scaling difficulties as the numbers of applications and repositories rise. At Johns Hopkins, we are promoting the idea that applications should access repositories through an abstract, repository agnostic layer, rather than through custom application to repository integrations. With funding from the Mellon Foundation, Johns Hopkins University will evaluate repository software and a range of services. The result of this evaluation will be a set of best practices, recommendations, and functional requirements for repositories and applications. This project reflects our belief that content should reside in multiple repositories external to applications, so that the same content can be used by several systems and support multiple services. This concept will be tested with content that is moved through repositories into applications as defined against a set of use cases that reflect various services. While our project will evaluate uses for digital preservation (e.g., LC Archive Ingest Handling Test), e-learning (e.g., Sakai), and e-publishing (e.g., Project Muse), this briefing will focus primarily on application to learning management systems.

The focus of this presentation is the integration of repositories and applications that provide services. It was encouraging to hear presentations from multiple repository and application development projects at a meeting at the Mellon Foundation last week that stressed the importance of integration through use cases (which is the approach we're proposing).

Perhaps a broader challenge would be to consider a perspective beyond the repository (which is arguably the library perspective) or an individual application (which is arguably the perspective of an educational technologist) looking at a range of uses, but this presentation will discuss e-learning in particular. The goal is to determine the architecture that will be necessary to combine multiple systems to serve a range of needs. Our primary consideration is how we actually MOVE CONTENT between these systems.

We start with the perspective of someone who supports e-learning. In this respect, you could begin with courseware.  With a certain perspective, this seems well contained in that one can offer the service and house the content within the application. This situation is depicted in Illustration 1.

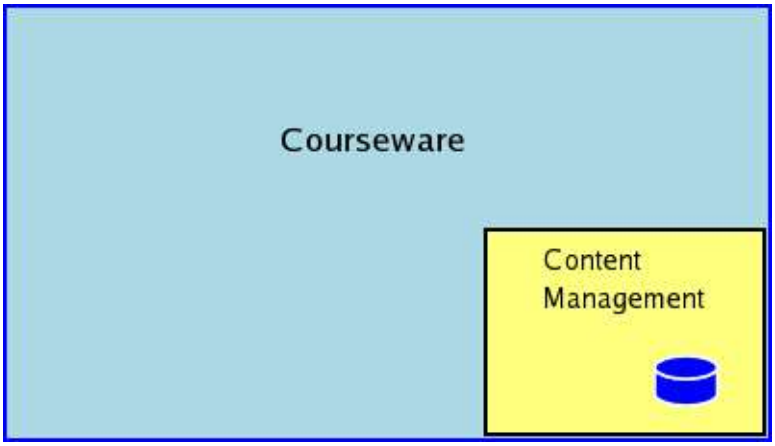


*Illustration 1*

*Text 1*

But what happens when you want to add functionality?  For example, there's a great deal of discussion about other tools being used for learning such as blogs, eportfolios, wikis ...

One approach would be to build additional modules or functionality onto this courseware application, perhaps similar to the “building blocks” approach shown in Illustration 2.

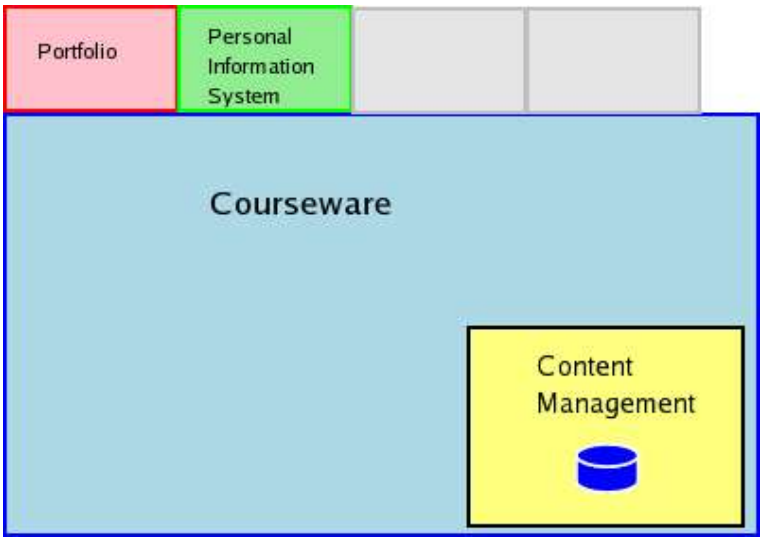


*Illustration 2*

Our premise is that this approach is problematic for a few reasons, most notably that a monolithic approach becomes somewhat unwieldy or bulky ("feature creep"). Another issue is that of lack of choice or flexibility, as one must choose vendor modules. What happens if we wish to use another eportfolio or library management system? A vendor may eventually support other such systems, but most probably on their own timeline, or according to their strategic or business plans.

We propose another approach that adopts open standards, modular approach to applications, and which does not prohibit use of vendor-based options, but which only prohibits using a monolithic, closed vendor-based approach.

Sakai is a collaboration and learning environment, OSP is an eportfolio application, Chandler is a personal information management application. These are open-source Mellon projects, but I don't mean to imply that these are the only choices. You can most probably identify an application that works in these areas (e.g., WebCT instead of Sakai)

Imagine that you wish to share content between these applications – for example, calendar or schedule information between Chandler and Sakai, or a personal, student essay between Sakai and OSP. (Illustration 3).

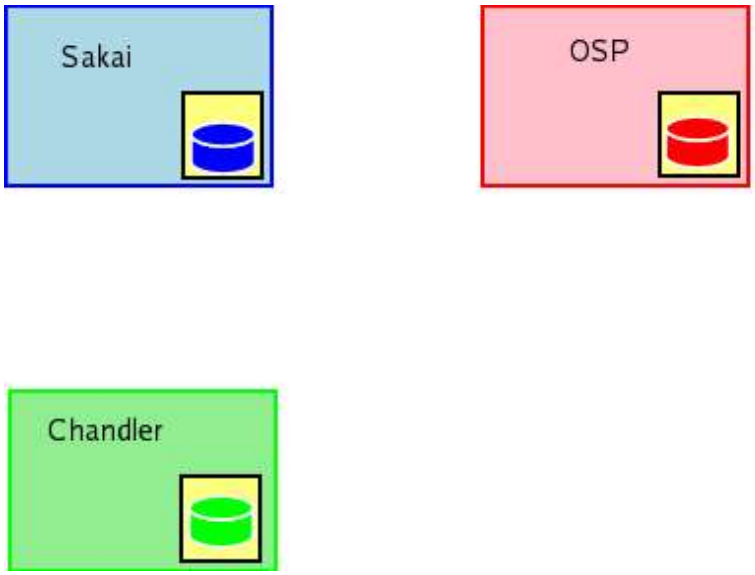


*Illustration 3*

Now consider connecting them to access the other systems's content (Illustration 4). As you can guess, these arrows are shown to depict these types of connections.

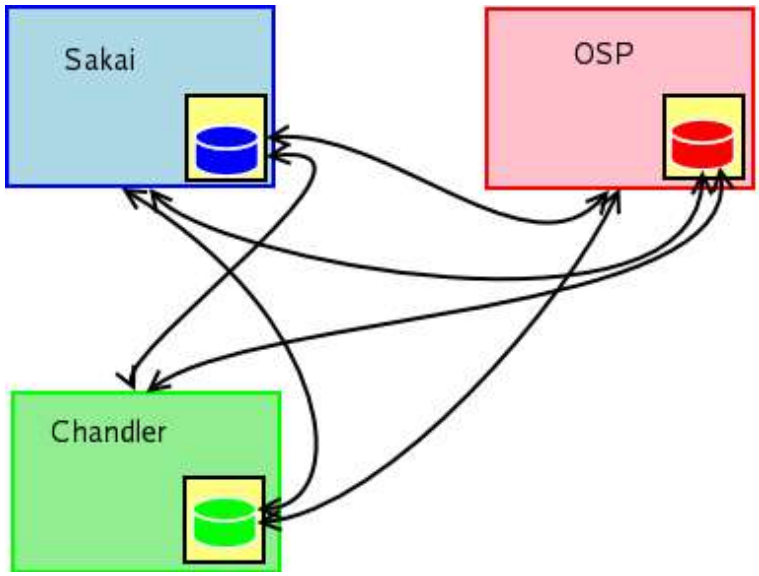


*Illustration 4*

Now imagine a situation with an even more complex scenario, that of using LionShare, which is a P2P application. In some sense, P2P applications sit somewhere between an application and a repository since it supports both services and storage. (Illustration 5)

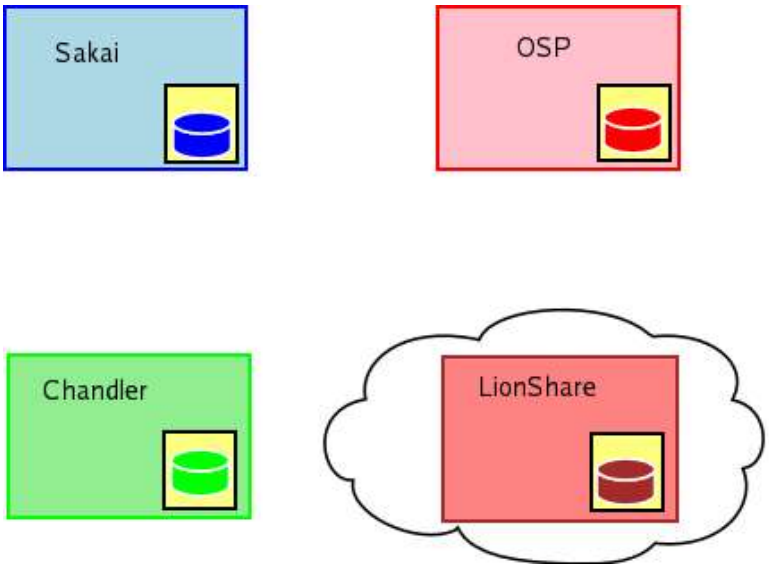


*Illustration 5*

Illustration 6 shows what happens when one considers the pairwise or bilateral connections between three applications and the P2P system.

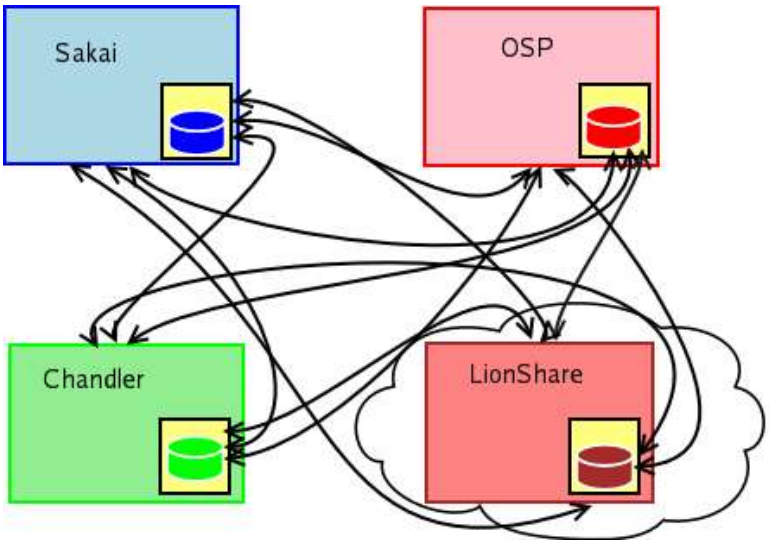


*Illustration 6*

You might recall that I proposed this open, modular approach to reduce complexity and unwieldiness, but looking at this diagram, you might be wondering about the benefits.

One way to manage this complexity is to remove the content storage aspects of these applications. By using a persistent repository to store the content and having the applications interface with the repository - which looks better. (Illustration 7)

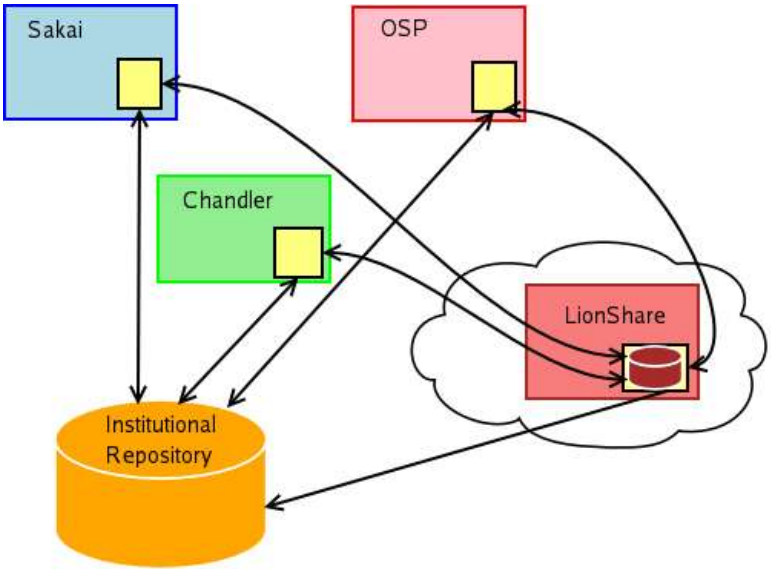


*Illustration 7*

I should also stress that one of the reasons why we believe content should reside in the repositories is that it increases our ability to support preservation. Storing content in generalized, non-fixed formats ("simple" or "dumb" repositories) gives us a better chance of handling that content over time. At this point, you might think that life is good – but just as there are many applications to consider, there are also many repositories to consider – see Illustration 8:

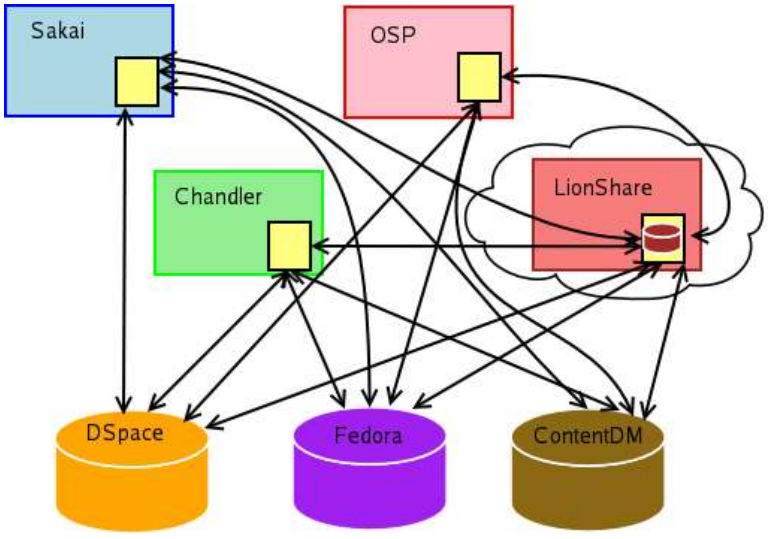


*Illustration 8*

I've chosen specific names to highlight that institutions are making different choices, but there are different types of content to consider. One way around this complexity is to create an interface layer to mediate connectivity between applications and repositories (Illustration 9).

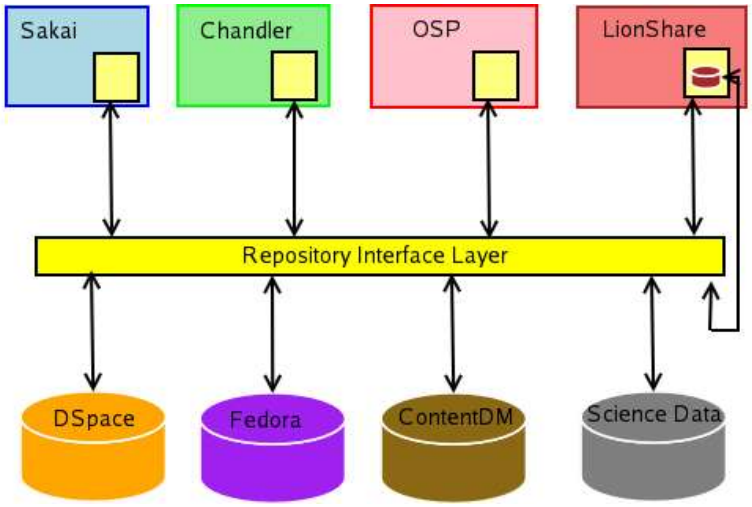


*Illustration 9*

That is, applications could write to this layer without knowing what repositories might access it. And repositories could write to this layer without knowing what applications will access it. By doing so, we take a (basically) m x n problem and make it a (basically) m + n problem.

It's one thing to have this diagram, it's another to get to our focal point of MOVING CONTENT through these systems according to this idea. In order to test this proposed approach, we intend to work with a series of use cases that will describe different ways in which people may wish to access content and modify it, repurpose it, etc. within applications.

I've added science data to this last slide to emphasize the growing repository problem, but I will mention in particular how important this area has become. At Hopkins, we've been working closely with the Virtual Observatory Project to consider their data curation needs. Use cases from existing proposal effort. In some cases, we've already gained some experience with ingesting content (through AIHT). In others, we're still developing use cases.

**Use Case Approach**

In the second half of this presentation we will talk about the approach we're adopting for use cases, with a particular emphasis on e-learning. The use case approach we are using is fairly straightforward, and involves four main steps.

Generally, we

- Start with scenarios, or “stories”
- Map the sequences of key events
- Group scenarios according to the project focus
- Develop use cases from classes of scenarios

The scenarios are simply descriptions of specific interactions between users and systems. We divide scenarios into groups according to how their sequences of events agree, so that many different scenarios will collapse into relatively few use cases.

We will keep track of the following information for our use cases, after Kulak and Guiney:

- Use Case Name
- Summary
- Actors
- Basic Course of Events
- Alternative Paths
- Exception Paths
- Extension Points
- Assumptions
- Preconditions
- Postconditions
- Author
- Date
- Scenarios

Most of these fields are self-explanatory. However, I think a word of clarification on the difference between a precondition and an assumption is necessary. A precondition is understood to be something inside the system under development which is necessary for the use case, but is outside the scope of the use case itself; assumptions are also necessary for the use case, but are outside the scope of the development project. The last field, scenarios, is just a list of references from the use cases to the various scenarios that comprise the use case.

Here is an example of a use case for use of images from the Visual Resource Center in our Art History department.

**VRC Use Case 2**

One professor using same images in different courses

Every semester, Professor Bilbao teaches an introductory photography course, and a course on the history of photography. She uses digital images to teach both courses, and frequently shows the same images to both classes. The current image database and presentation software that she uses (MDID v.1) allows her to search for images and save them as slide shows in a course folder, which she can present in class, and the students can review on-line. The folders for the courses, however, are completely separate, so she must create a slide show in a specific course "folder," and recreate the same slide show for the other course in its course folder. It would be a lot easier if she could create the slide show only once, and put it in any course folder she needs to. Also, she likes to include comments about images in the slide show, but these comments vary, depending on the course in which she uses the image. For example, Edward Weston's *Pepper No. 30,* discussed in the Introductory Photography, may be important for illustrating depth of field, but important in the History of Photography as an abstract work of art.

Here is a possible sequence of key events for this scenario (Illustration 10):

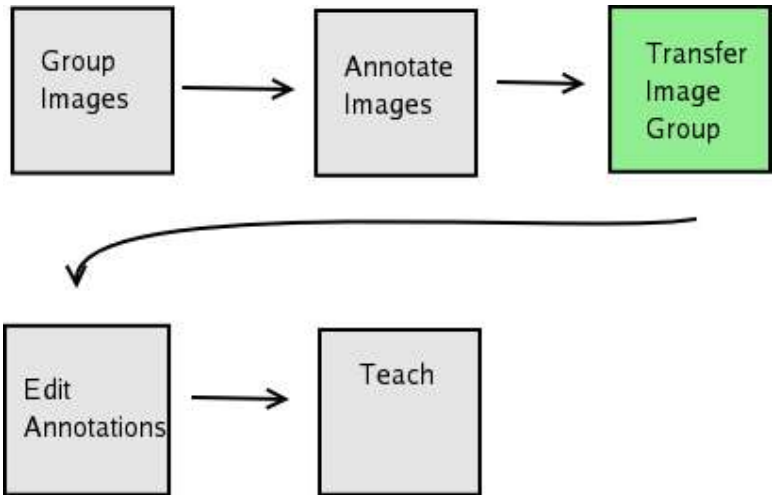


*Illustration 10*

Here is the use case which corresponds to this scenario (some empty fields omitted):

| Use Case Name: | **Show same set of digital images in two different courses** |
|---|---|
| Summary: | Professor uses a set of digital images in two different courses that s/he is teaching. What s/he has to say about the images is different in the two courses. |
| Actors: | Professor |
| Basic Course of Events: | 1) Professor groups images by lecture topic for one course.<br>2) Professor annotates images with notes particular to that course.<br>3) Professor transfers image groups to the other course (in one easy step, not image by image).<br>4) Professor edits the annotations so that they are appropriate for the second course.<br>5) Professor teaches both courses, using the same image groups but using them to illustrate different points. |
| Extension Points: | |
| Triggers: | Professor is asked to teach two courses where the visual material can be similar. |
| Preconditions: | Professor has a set of digital images. |
| Author: | Meghan Gross, Teal Anderson |
| Date: | Created 2004-12-21. Last modified 2005-03-10. |
| Scenarios: | VRC, Use Case 2, Introductory Photography and History of Photography courses |

We note that the same scenario might be interpreted differently to give different sequences of key events (Illustration 11).

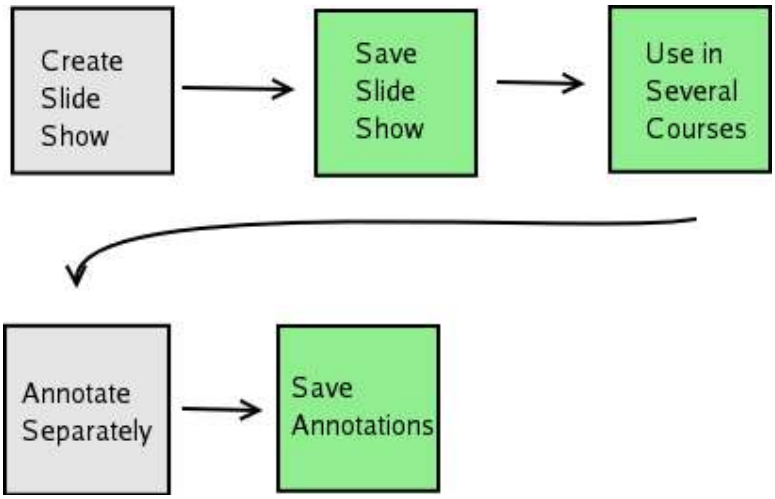


*Illustration 11*

If this is the sequence of key events, then the use case would change accordingly.

**Applying Use Cases**

In our project, we will derive functional requirements from the use cases, and then investigate support for these requirements in both repository systems and existing repository interface specifications. We can then compare support across use cases, repositories and specifications to see find gaps which would obstruct the use of various combinations of repositories and interfaces to support the use cases.

For the sake of simplicity, suppose we end up with four different use cases, and identify a total of ten different functionalities needed to support them. We could form a matrix to show which use cases required which functions, and also which functions are supported by the various repositories and interfaces. (In all of the following tables, green cells indicate that the function labeling the row is required or supported by the item which labels the column; otherwise, the cell is red). For the sake of argument, let's suppose that our use case functionality requirements turn out as follows:

| | ***Use Case 1*** | ***Use Case 2*** | ***Use Case 3*** | ***Use Case 4*** |
|---|---|---|---|---|
| Function 1 | | | | |
| Function 2 | | | | |
| Function 3 | | | | |
| Function 4 | | | | |
| Function 5 | | | | |
| Function 6 | | | | |
| Function 7 | | | | |
| Function 8 | | | | |
| Function 9 | | | | |
| Function 10 | | | | |

Similarly, we can investigate which of these functions are supported by various repository interfaces.

| | ***JSR 170*** | ***OKI DR OSIDs*** | ***IMS DRI*** |
|---|---|---|---|
| Function 1 | | | |
| Function 2 | | | |
| Function 3 | | | |
| Function 4 | | | |
| Function 5 | | | |
| Function 6 | | | |
| Function 7 | | | |
| Function 8 | | | |
| Function 9 | | | |
| Function 10 | | | |

Next we can form the same matrix, this time looking at repository systems to see which functions they support:

| | ***DSpace*** | ***Fedora*** | ***Content DM*** |
|---|---|---|---|
| Function 1 | | | |
| Function 2 | | | |
| Function 3 | | | |
| Function 4 | | | |
| Function 5 | | | |
| Function 6 | | | |
| Function 7 | | | |
| Function 8 | | | |
| Function 9 | | | |
| Function 10 | | | |

Putting all this information into the same table will allow us to see at a glance which interface/repository combinations support the various use cases. For example, Use Case 1 can use any of the three repositories and any of the three interfaces, since its required functions are just the ones numbered1-3. Use Case 2 requires functions 3,7,8,9 and 10, and as such is only supported (in this example) by the JSR-170 interface and the Fedora repository.

| | ***UC1*** | ***UC2*** | ***UC3*** | ***UC4*** | ***170*** | ***OKI*** | ***IMS*** | ***DS*** | ***FED*** | ***CDM*** |
|---|---|---|---|---|---|---|---|---|---|---|
| Function 1 | | | | | | | | | | |
| Function 2 | | | | | | | | | | |
| Function 3 | | | | | | | | | | |
| Function 4 | | | | | | | | | | |
| Function 5 | | | | | | | | | | |
| Function 6 | | | | | | | | | | |
| Function 7 | | | | | | | | | | |
| Function 8 | | | | | | | | | | |
| Function 9 | | | | | | | | | | |
| Function 10 | | | | | | | | | | |

More importantly, we will also be able to identify which further functionalities will need to be supported in both the interfaces and the repositories to enable additional services as defined in the use cases.