Prepare the evolution

Moving towards a service-oriented architecture

ELAG Conference, Barcelona 9-11 May 2007
Prepared by: Erlend Gutteberg, System Developer

Background

At BIBSYS there is a long tradition in exposing services onto internet. This is mainly done through standard interfaces like Z39.50,OpenURL and OAI. In many ways this could be seen as a web service, and per definition part of a service orientated architecture. So why the sudden buzz around service oriented architecture, or SOA as it is known? What can this bring to the library world?

What do we know about the future of the library and the library automation system?

- Based on hive mind metadata, driven by user contributed information?
- Constantly changing in a radical way?
- Interact with other systems in much more complex workflow's?
- Use search and retrieval methods we haven't even thought of yet?

All this may or may not be true. What we know is that the library will change, and that it will change whether the automation system does or don´t.

The scope

In light of what to come, BIBSYS started it's "Ready to serve" project, early 2006. This is a project implemented in cooperation, and partly funded by the Norwegian Archive, Library and Museum Authority

The scope of this project is;

- Conversion to and from the internal BIBSYS MARC to MARC21 metadata format, and a service to expose this publicly.
- Metadata-delivery through OAI,SRU and SRW services.

The development team dedicated to this also worked on parallel projects with indirect influence on the "Ready to serve" project. In this way the result has been a much more thorough implementation than the initial resources would imply.

Analyzing the possibilities

When starting out, there where some key considerations to be taken. Due to the uncertainty regarding the future of libraries in general. One of the goals would be preparing the system to rapidly adapt when needed. Also, on the way, use modern and standard software, or prepare for easy replacement of structural components. Such as the persistency layer of the library system.

All the required services could be implemented through a simple kind of wrapper on top of the old system. Why bother to go further? If we knew there wasn't going to be any changes to the library system, it would be an easy decision. The obvious, would be a quick implementation. But taken in regard, the expected change rate of library systems, and information systems in general. This would, at least, double the workload when seen in a longer perspective. And Increase complexity in an already over-complicated system.

During this process it become clear, that only defining external services wouldn't accommodate flexibility enough to handle the future demands to the library and it´s automation system. What if we could divide the system in atomic modules, only separated by explicitly defined contracts and messages? We then would be able to rapidly put together new services, and mash up old ones. Maybe even reuse some services, and get more out of the effort put into it.

The philosophy

The key philosophy to reach our goal on flexibility is “Decoupling”. By decoupling we mean, making the software components as independent of each other as possible. When components are decoupled, the idea is that they will fit much better in a service orientated architecture. The components will be (more) easily replaced by new or alternative components. They will generally be of a more general purpose nature, and conceptually easier to understand, by both information systems and the people designing them.

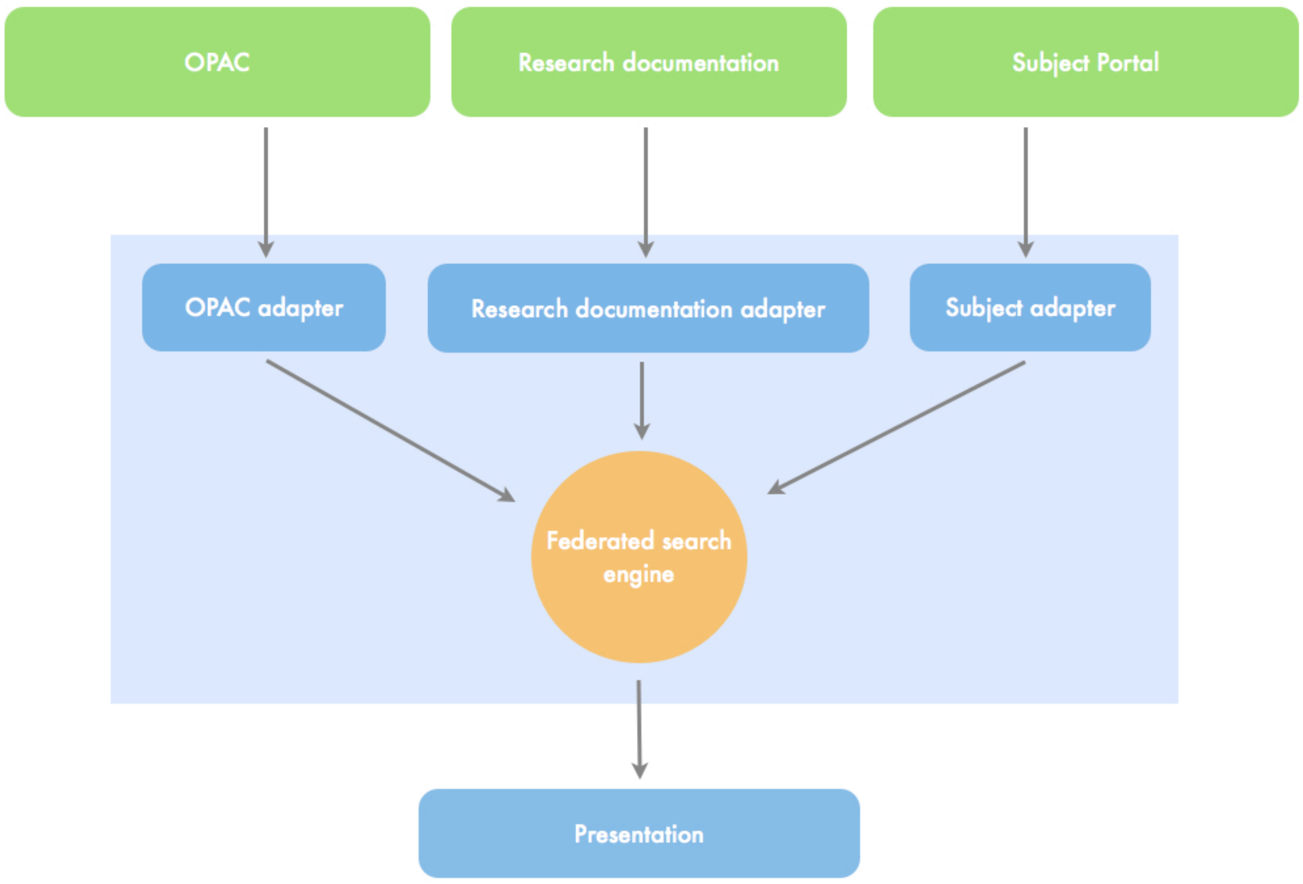
The work

The in-house developed federated search engine Ask (http://ask.bibsys.no/) worked as a basis, when redesigning the search and retrieve service. Ask searches through these BIBSYS services;

- BIBSYS library catalogue
- BIBSYS research documentation catalogue
- BIBSYS subject portal

This was done by making an adapter for each of the searchable databases (producers), as shown in illustration 1.

Illustration 1, The federated search engine, based on adapters.



When working with this model, we recognised that there where huge amount of code in the adapters, and that this was duplicated from the producer systems. To avoid this, and decouple the search engine from the producing systems, the presentation layer, was moved to the producer it self. This as shown in illustration 2.

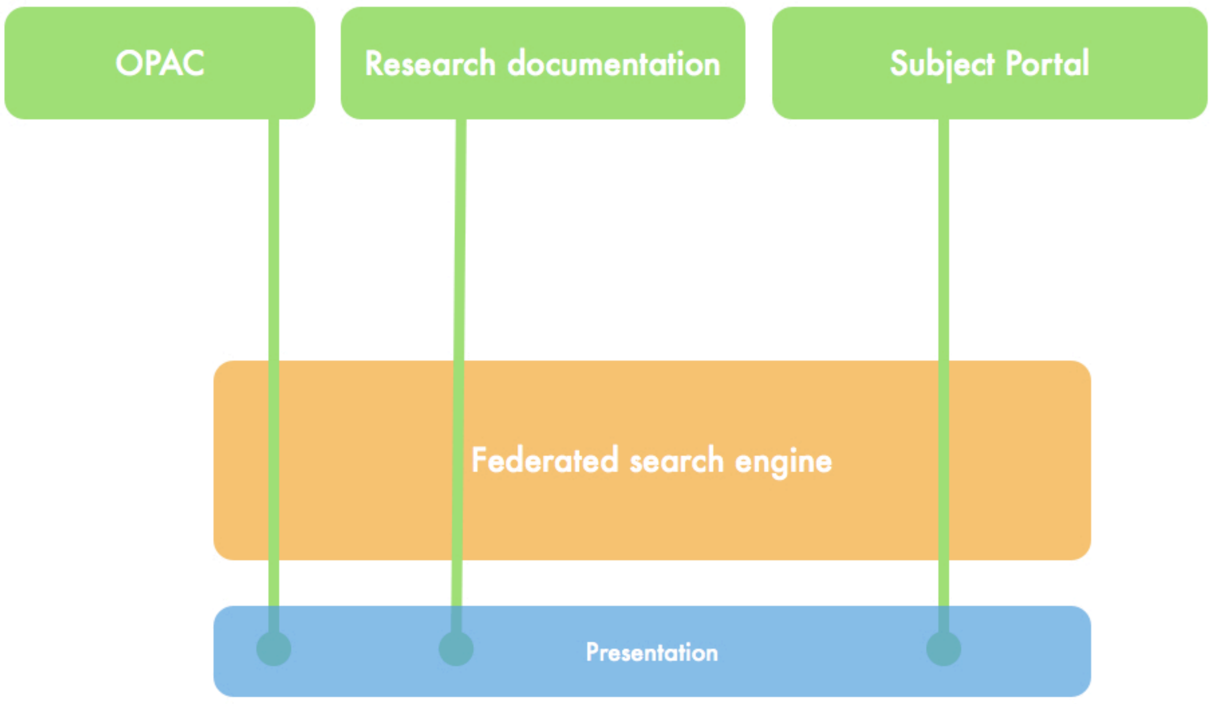
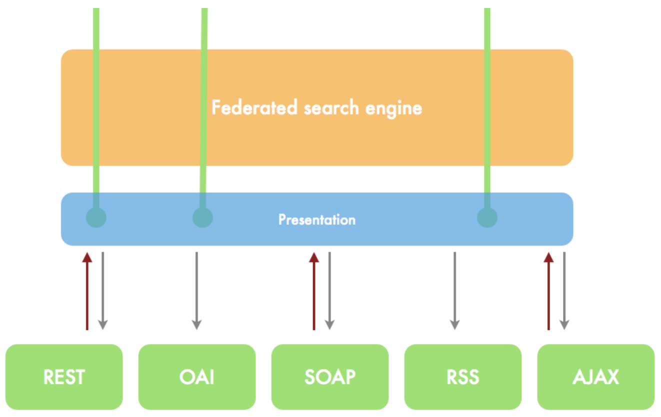


Illustration 2. The federated search engine decoupled and it´s decoupled producer systems.

By generalizing interfaces and remove dependencias, we end up with a wast variety of commonly usable services. Any supported metadata format may now be converted to another, and presented as OAI, AJAX,REST,SOAP and so on.



This is all centered around pulling information from the different systems. Another project, in cooperation with The Nation Library of Norway, extended the capabilities even further. By allowing update of metadata through the same architecture, wich have only been done through the OPAC, marks a turn in BIBSYS frameworks and architecture.

So the current characteristics of this architecture is;

- Multiple information sources
- General metadata conversion
- “SOA ready” architecture
- Search information
- Retrieve information
- Update information
- Plug-able presentation

While the scope of the current “Ready to serve” project, is limited to a small set of services, the framework that come out of it has a far larger potential. We are currently working on enabling the new architecture on Ask (http://ask.bibsys.no), our public accessible search engine. At first this will give little new functionality, but prepare for the changes to come. Making it an adaptable and easy integrated application.

The next step

One interesting aspect is that even though we say the architecture is service orientated, it is still bound through local interfaces, and mostly run on the same machine. We have reduced dependencies and made conceptual atomic components. The reason for this, is to decouple also the binding mechanism. This in such a way that even the communication protocol between the components is unimportant of the functionality. By doing this, the system can scale by distribution, run on different kind of architectures, and generally evolve naturally while software technology change.

The conclusion

This project is very much a work in progress. We see endless possibilities in both including more service-resources on the service layer, but more importantly making it possible for 3rd parties to reuse our services in a way they find both easy and comfortable. Services like RSS feeds and Javascript-friendly feeds will be natural to implement on such an environment.

One of our goals is to design systems as part of a larger workflow, exposing services to other systems outside of BIBSYS. This may be implemented as a full layer of SOAP services. Potentially open our systems on a lower level, giving application functionality like ordering books and updating metadata.

References

Fowler, M. (2004) “Inversion of Control Containers and the Dependency Injection pattern” (http://www.martinfowler.com/articles/injection.html)

Angerer, B and Erlacher, A (2005) “Loosely Coupled Communication and Coordination in Next-Generation Java Middleware” (http://today.java.net/pub/a/today/2005/06/03/loose.html)