The Fascinator @ !dea MMVIII

The Fascinator is a bit of repository software, designed to show off stuff that resides in a Fedora Commons store. It was designed and built at the Australian Digital Futures Institute at The University of Southern Queensland (USQ). The software development was funded by the ARROW project with a contribution from USQ. It is based on some work that was part of FRED (Federated Repositories for Education). None of it, including the ARROW software would be possible without the Free and open source software on which it was built and in particular the indexer for Fedora that the Muradora team adapted.

The Fascinator was largely written by Oliver Lucido, with task management and web-skinning by Bron Chandler, and some programming on the harvesting component from Tim McCallum, not to mention the screencast. Neil Dickson at ARROW was our chief stakeholder. Also holding a stake was Alison Dellit from the National Library of Australia (NLA).

Why was The Fascinator relevant to the audience at !dea? The motto there was Enabling Technology: Forging a Digital Future – it was a cross-sectoral meeting for educational technologists (at least that's what I thought it was).

I had to think about that a bit. In addition to covering the software I tried to explore these themes:

1. Talk about some contrasting software development/procurement approaches.
2. Look at standards and interop in Higher Education, particularly how the higher-ed sector has gone trying to harmonize standardize and align their metadata.
3. Is the e-Framework any use to a dev shop like ours? (If you don't know what that is, don't worry, the e-Framework crowd can comment below if they like)
4. Open Source as a conversation.

To understand the motivation for The Fascinator we have to look a little at the ARROW project. ARROW (Australian Research Repositories Online to the World) was (it ends at the end of 2008) an Australian government funded project. ARROW got involved in an interesting software development model, partnering with a commercial systems vendor from the USA to build some Institutional Repository software on top of an open source base.

The result was an, um, *interesting* mix of free and commercial software.

There was some talk on Thursday morning about building vs buying technology. The ARROW-sponsored application was written as part of a complex interaction in which some technology had been bought but some was open, and we built something that complemented the bought bit and filled in some gaps. That's much more complex that just 'build or buy'.

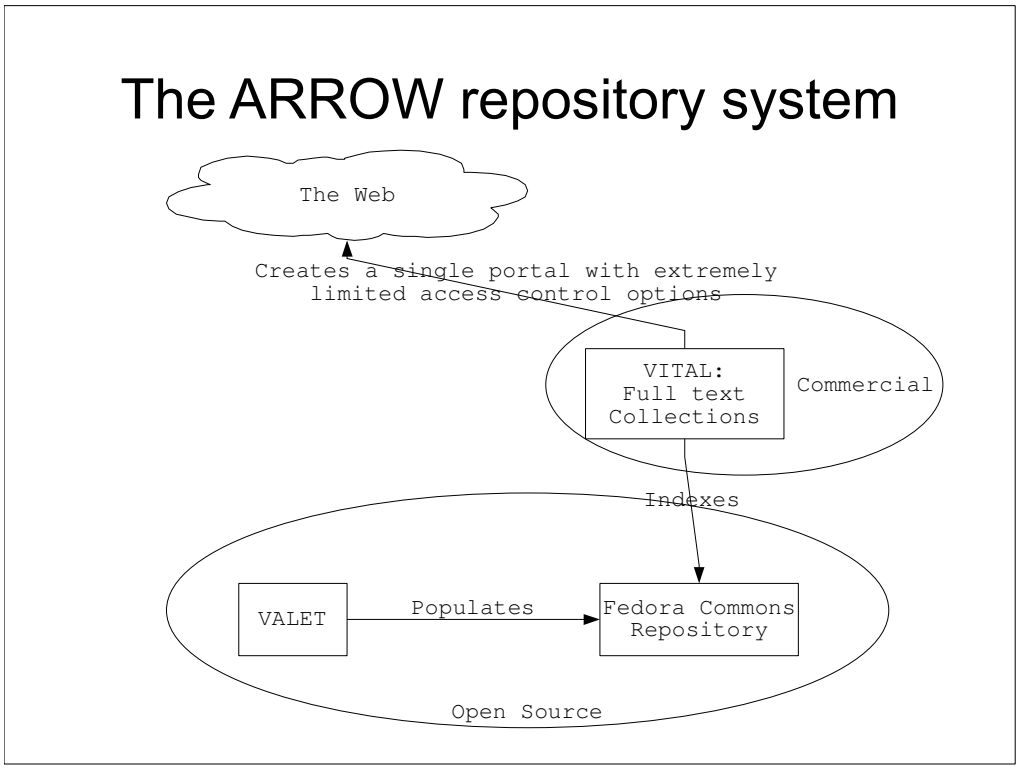
I 'm not up to expressing this in 'proper' eFramework-speak, but broadly speaking the services the ARROW community are after include:

1. Collection management; the ability to browse and search groups of like or related things.

2. Full text search.
3. Simple access control such as on-campus access only to some kinds of document, rather than more complex far-fetched requirements.
4. Separate views or 'portals' for different campuses or purposes. One thing that would be nice would be to separate images from documents for example. Can I have a document portal and an image portal so I don't keep seeing pictures of cathedrals when I want to read theses?

(Apparently one of the eFrameworkistas was working on a Service Usage Model as I spoke. Can I see it please?)

The ARROW software, which is called VITAL, currently does 1 and 2 above, and has very rudimentary support for 3, while 4 is a feature that is coming in the new not yet released version 4 of VITAL.



While the VITAL software in the above diagram is represented as a nice clean box, what goes on inside is probably rather complicated. Of course it's not possible to find out exactly what VITAL does as it is a proprietary closed-source product. What I know about VITAL comes from talking to people from the vendor. I don't want to get into too many specifics as I know the product is evolving and they might not like me to share all of the things they have shared with me. I have based this discussion on knowledge that is common to in the ARROW community.

Now the current version of VITAL can't fulfill all the requirements I outlined above, but lets think about a imaginary repository at an imaginary university – the

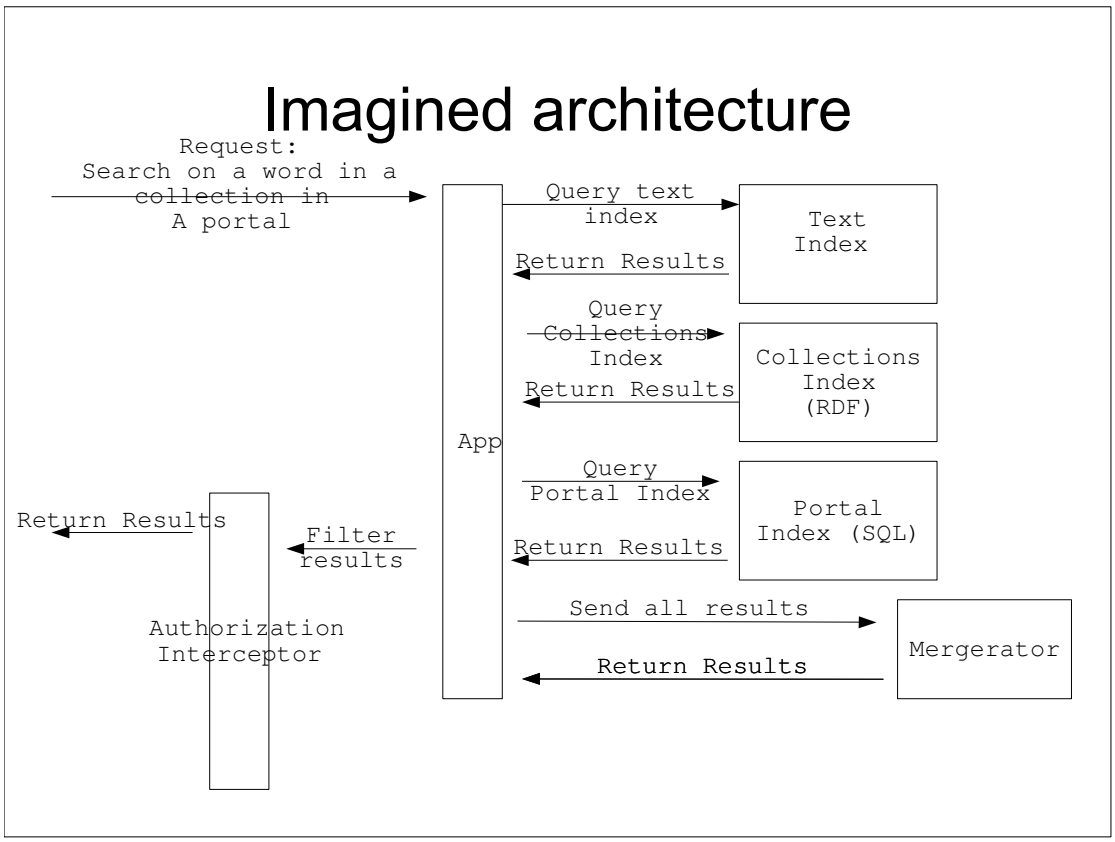
University of the East Coast. That institution might have a repository which has multiple campuses (Bondi Beach, Surfers Paradise, North Stradbroke Island, etc), and which has 'collections' of documents such as research articles or theses. Some of the material is not for public consumption, such as theses which are under embargo because they contain stuff that's patentable.

This diagram shows what I imagine might go in inside a repository as it grinds through the process of doing a search. This involves doing four separate lookups. I have no idea how VITAL really works so please don't take this literally – it's a thought experiment.

If I search for something, say the word “surfboard” in the thesis collection, from the portal belonging to one of the campuses then:

First it must merge the results from three of its indexes;

1. A text index.
2. A collections index using the RDF triple-store.
3. And a portal table which might be implemented using a SQL database.
4. When that's done it has to compare all the results to the access policy store in a process know as post-filtering.



When I gave this talk an audience member wanted to know why I wanted to post-filter search results. Answer? I don't. This slide was to illustrate the

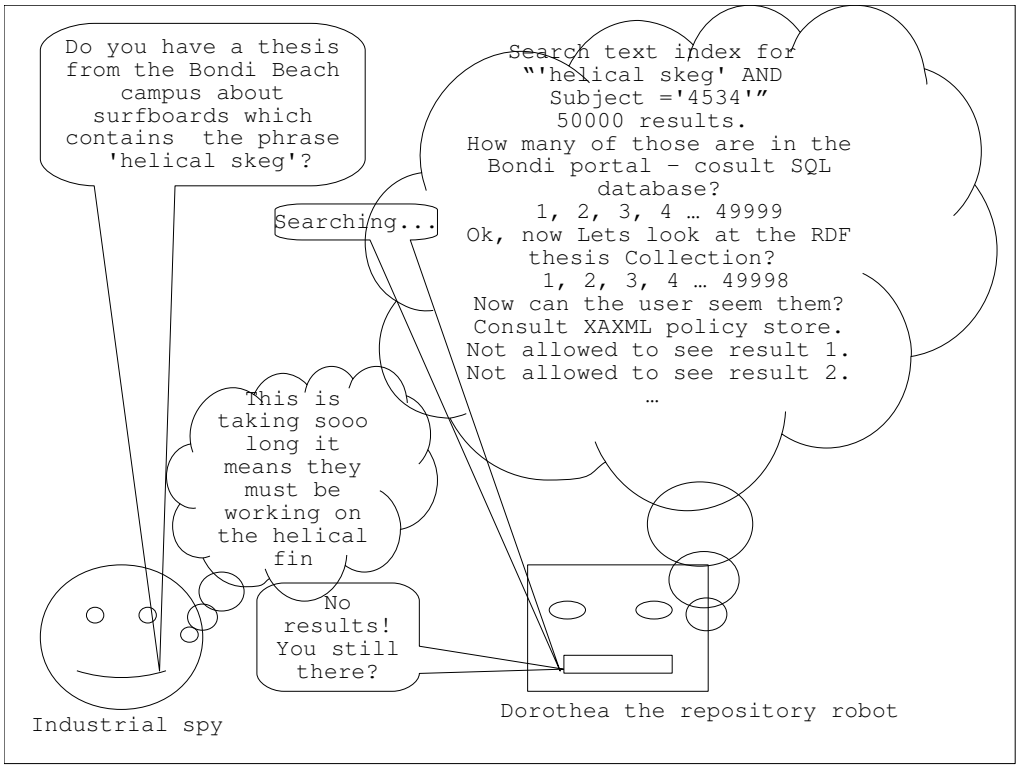
problem with this kind of architecture. Got that? I don't think that the above is a very good idea unless you want to spend a lot of time and resources on optimizing out all the problems. It's an anti-pattern.

This example illustrates a potential problem with using an architecture with multiple different indexes where queries need to be merged. The application designers will either have to accept slow performance or write a lot of optimizations themselves to merge all those results (using the mergerator which is a word I just made up) from searches against different indexes. This is complex, hard stuff.

The main problem with this architecture is post-filtering search results for access control. The impact of having to do this could be very great in some situations. I have concocted a ludicrous example to illustrate this.

At the University of the East Coast, there's this rumour going around that someone is onto a new kind of skeg with a special shape. In fact there is a (very, stupidly,) large number of PhD theses that have been written on the topic but they are all under embargo and the people who wrote them are sworn to secrecy.

So when a designer from a surf company searches the uni repository it's important not to reveal that, yes in fact we do have a lot of hits for the search term 'helical skeg'.



You can tell that I made this example up, right? But it does illustrate an important point. If you have some non open access content you need to be very sure that you do not expose even the fact that you have a document that contains a particular string.

This illustrates only what I see as a potential problem with using an

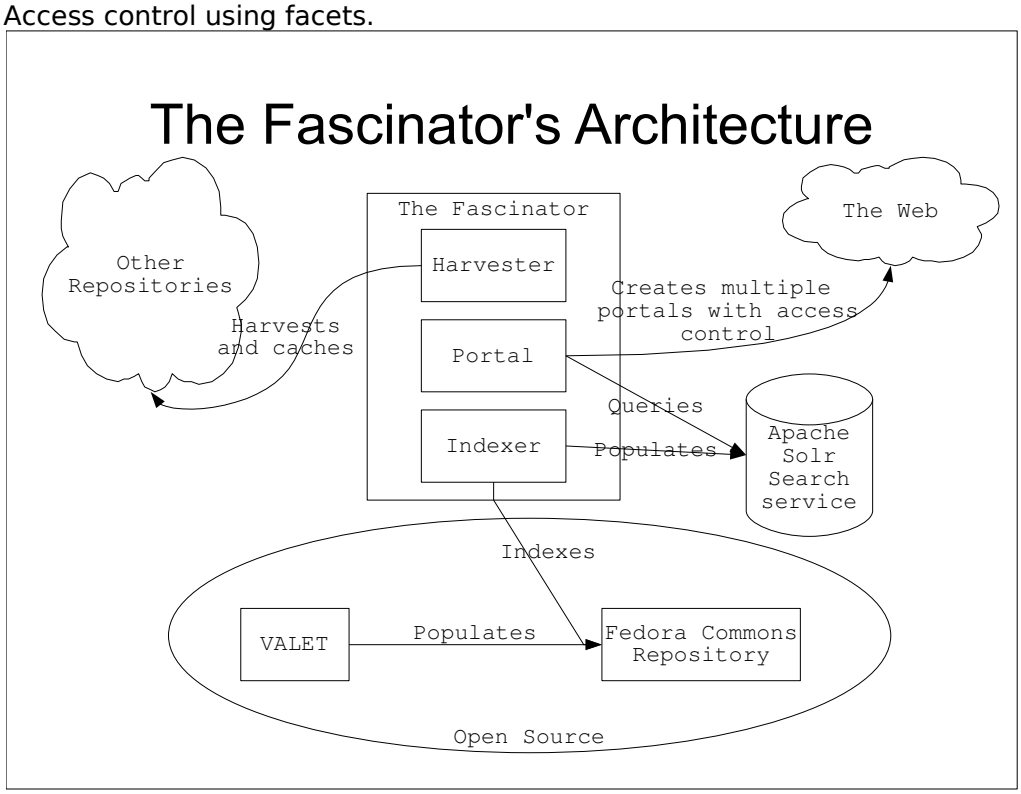
architecture with multiple different indexes where queries need to be merged. The application designers will either have to accept slow performance or write a lot of optimizations themselves. This is complex, hard stuff.

The Fascinator takes a different approach; it uses just one technology to the drive the search; Apache Solr. Want to do a text search? Apache Solr. Limit results by campus? Apache Solr. Limit the amount of stuff guests can see? Apache Solr.

One big advantage of this approach is that Solr already handles lots of the internal caching of results. If you search for everything that guests can see it remembers the result. If you search for everything with an index entry of `Collection='thesis'` it can remember that. When you perform a complex query smart programmers have already worked out the fastest way to merge all the pre-cached results. It's built in.

Main features:

1. Flexible indexing.
2. Faceted search.
3. New portals using facets.
4. Access control using facets.



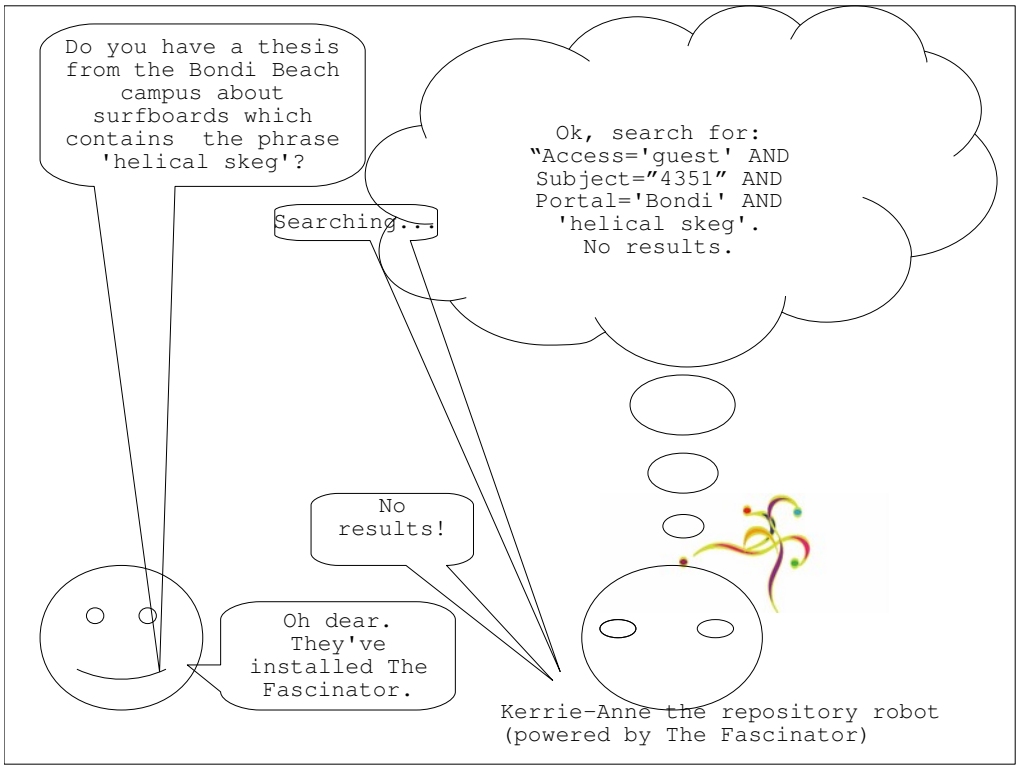
You can see the Fascinator in action in a couple of places:

- Our demo site. Currently has content from a variety of places including the Université catholique de Louvain where we have harvest open content for

demonstration purposes but your mileage may vary.

- The AURIC site where we will eventually harvest all the Australian University Repositories we can get our paws on.

The Fascinator is mission-accomplished in terms of its goals, which were really about making a proof-of-concept statement in an ongoing conversation with the vendor and others in our repository community. We could park it now and be happy – but that would be a pity. It looks like the software could have a life beyond the initial project; there has been interest expressed in a quite a few places, but everyone involved needs to understand that support from now on is based on whatever we can manage in between other projects, unless we can work out a way to make it sustainable.



There are a couple of things we can do now:

- Use it as a simple portal for Fedora as part of a simple repository along with the new Java version of VALET we're writing.
- Explore how it might fit with other software, such as ePrints which has a highly developed workflow for ingesting research outputs, or Muradora which specializes in access control. There's no reason we couldn't make The Fascinator work nicely as complements to these and more.
- See how it will work in the new (to us) world of eResearch data that gets created in the lab or in the field and then must find its way into long term storage and global sharing. I'm thinking of a sort of 'iTunes for research data and publishing' but without the lockin of course.
- Consider The Fascinator as the basis for a crowd-sourced history project with

Chris Lee of the Public Memory Research Centre at USQ (he just got funding).

- Think about how we might index RDF, as there are lots of projects starting to think RDF in the eResearch world.

In any case we will offer the indexing component, which we (meaning Oliver) built on top of work by the Muradora team back to the Fedora Commons community.

In conclusion, I offered the folks at !dea at 2008 the following. (Don't think I really offered much re software development methodology):

1. I observed that from what I've heard about the vocational (VET) sector they have done much better with metadata standardization. From what I hear there was a bit of cynicism from some people about that, but on the whole I think that's right – they're more focussed and have done much better at aggregation and sharing. See LORN.
2. I'm still not sure how I would use the e-Framework to design software – what we did here feels more like improvisation; mashing up a couple well-known patterns. Sort of like playing a Reggae song using country licks, which is what I do at home as my hobby. But talking to Lyle Winton, he says that what we have done is identify an anti-pattern (the “imagined architecture” above) and a new pattern which works. The e-Framework team are welcome to use us as a case study...
3. This open source thing has been really important to us. Instead of being stuck in conference rooms and conference calls trying to explain to a software vendor what ARROW wants, we were able to spend a comparatively small amount of money and say it with code, building on what others have already done.