OOR Federation

Dan Cerys
Jim Chatigny
Mike Dean

Panel Session:
Coordinating our OOR Software Development

19 February 2010
• OOR Federation allows metadata that is hosted remotely to be browsed locally.

• Our implementation is patterned after the “OBO Pull” capability.
  – Runs periodically via the Quartz Scheduler.
  – Retrieves metadata from remote hosts using their RESTful services.
  – Synchronizes local copies using the BioPortal core management functions.
  – Ignores remote metadata that is itself federated.

• The initial version just supports Ontologies and their Views.
  – Other RESTful services could be exploited (e.g. diffs, metrics).
  – We may want to add a RESTful service for mappings.
  – The requirements for other items such as users and projects are unclear.
    • Might need to employ a unique global identification scheme?

• The software resides in our Subversion repository, extended from:
  – bioportalui: branch OOR, rev 434
  – bioportal_core: tag 1023, rev 2405
  – bioportal_admin: rev 120
Our server is federated with the cim3 OOR sandbox.
Federated Ontology Example

- Federation creates local copies of Ontology Beans and Files.
- Editing of local copies is disallowed.
- But links to the remote OOR hosts are provided.
All versions of Ontologies and Views are supported.
Federation is also available via the Admin app.
• Added new federation configuration settings.
  - oor_federation.xml.tmpl: for remote host specifications.
  - build.properties: for resource path and Quartz options.

• Added properties to the OWL metadata model.
  - federatedUrl: Base URL of the remote RESTful services.
  - federatedId: ID attribute of the remote OntologyBean.

• Updated the Java/Ruby models and their dependencies accordingly.

• Integrated new service classes into bioportal_core.
  - OORFederationService: performs the federation operation.
  - OORFederationRestlet: allows federation to be invoked via a URL.

• Modified bioportalui to include federation info and to disallow editing.

• Added a federation utility to the bioportal_admin webapp.
Managing State

- State is represented as sets of OntologyBean instances.
  - A separate bean instance represents each ontology/view version.
- Beans are organized into “synchronizer” objects specific to each remote host.
  - Local state = all local beans that are federated from that host.
  - Remote state = all beans residing at that host that are not themselves federated.

Processing Steps

1. Synchronizers are populated to reflect the local and remote states.
   - If communicating with a remote host produces an error, it is skipped.
2. Local beans are deleted if:
   - Their remote host is not declared in the current configuration.
   - Their remote bean is found to no longer exist.
3. User-editable fields of the remaining local beans are updated as necessary.
4. New local beans are created for each new remote bean that was retrieved.
Recommendations

• We should merge our sources into the group’s subversion repository.

• The group should more fully develop operational requirements.
  – Start with federation Use Cases?

• RESTful services should be implemented for all metadata artifacts that are deemed appropriate for federation.
  – And incorporated into the OWL metadata model?