RichardBoston2006PreliminaryNotes

Architecture Review Preparatory Notes

Some Issues, and their relevance/importance at this stage

These issues have been harvested from the various email discussions prior to the meeting. I have aggregated them and given them some ratings on how I feel they should contribute to the meeting.

X = Key discussion point
+ = Useful discussion point
~ = Indifferent discussion point
? = undecided

- Interoperability (X)
- Preservation (X)
- Format Registries ?
- Modularisation/Extensibility (X)
- Add On Mechanisn (X)
- User Interface in General (X)
- Scalability (X)
- Storage Layer (X)
- Content types (~)
- Persistent Identification (X)
- Complex Objects + related to Interop?
- Federation (X)
- Versioning (X)
- Web Services +
- Delegation of administration +
- Authentication and Authorisation +
- Internationalisation (~)
- Metadata architecture (X)
- Grid technologies (~)
- Data replication (~) this will be a by-product of other initiatives
- Workflow design +
- Integration (for other systems) +
- Disaster Recovery (~)
- Searching (~)
- Automated metadata creation (~)
- Maven/Ant +
- Usage Statistics (~) implementation specific-database by-product
- Database abstraction (X)
- Configuration (X)
- Unit Testing +
- Update path/process (~)
- AIP and OAIS +
- Structure Mapping + related to Interop?
- Collaborative workspaces (~)
- OAI-PMH +
- Exception Management/Architecture +
- Lifecycle Issues ?
- EPeople (~)
- Authority control of data (where is it?) (X)
- Test Data Set (~)
- RDF + SIMILE ?
- Provenance/Audit/History +
- Community documentation (~)
- System documentation (X)
- Technical Integration with the Community +

Mini-notes on thoughts that I also had

These are random jottings put down while preparing for the meeting

- OSGi
- Internal Object Metamodel: consider perhaps an internal description metadata for which we strongly specify the schema for, whose job is to describe /for DSpace/ what each of the Datastreams in the Item do, and allow that model to make it possible to load relevant content handlers dynamically, etc.
- Separation of identifier management (so we can easily use things other than handles)
- Support hierarchical metadata
- Java 1.5
- Interoperability has some layering: protocol interop, object structure interop, content interop. What is our scope? What are the issues?
- Metadata issues: FRBR?, multiple records per Item.

- All objects (e.g. Communities and Collections) require a proper metadata handling process as Items currently do (but with associated improvements for hierarchical schemas)
- Need to know more about Cocoon to understand at what stage in the processing the XML for the UI is produced. Presumably there is a strong separation of concerns for this processing and sub-user-interface layer.

Key issues

These are Key Issues as they have been presented in various emails leading up to this meeting, and have been mildly filtered for this list to reflect the things I think are important.

- What is in/out of scope
- Information model
- "Core" vs "Module" (plus, Kernel, UI, Backend)
- Interfaces
- Authentication and Authorisation
- Interoperability
- Scalability

Pre-reading notes

DSpace+2.0 Architecture

- I think that I need more information about Cocoon. Can it fulfill the part laid out for it in the "4 layer architecture"?
- AIP Formats: All content objects (communities, collections, items etc.) need to be represented in the same (or similar) way. An aside to this is that they should all also, therefore, be behind a proper interface.
- Push/Pull: Messaging framework; semi-synchronous; threaded core. Built correctly, the threading code can remove the need to understand the threading from the local developer. A variety of simple techniques exist to ensure that the content object and it's access cache are kept up to date with a reasonable delay.
- Modules: modules running from the top to the bottom of the stack look difficult, but it would be good to have new modules being reflected in the UI in some appropriate manner (again, does Cocoon fulfill this role?).
- Default distribution: The first instance of DSpace+2.0 must have at least as much functionality as the 1.x branch, and must be set up in a sufficiently similar manner.
- Authentication & Authorisation: Pluggable: native, LDAP, Shibboleth. Access control to objects should be done via something like a Proxy pattern.

DSpace White Paper

- "Don't be afraid to throw away unneeded code".
- Some broad areas of responsibility, drawn from this document:
    - Developer tools
    - Support
    - Development
    - bug fixing
    - documentation
    - administration
- Change "Bitstream" to "Datastream"
- Better access identifiers all round, IMO, including semantically meaningful URLs
- Bundles to be replaced by appropriate relational metadata
- Communities and Collections: these are semantically different, but do they really need to be separate objects? How about a "Collection" whose content types can be defined?
- Versioning: a complex problem which we must consider separately. See for example, the VERSIONS project: http://www.jisc.ac.uk/whatwedo/programmes/programme_digital_repositories/project_versions.aspx

Bitstreams Relationships

- This is really talking about structure mapping. It goes beyond bitstream relationships and into the realm of the complex object representation. Lets have a general solution, not another class to manage a specific relationship.