ArchReviewNotes

Contents

DSpace Architectural Review

w/o 23 Oct 2006

I. attendees

mackenzie smith* mit libs
john erickson* hplabs
john mark ockerboom* penn
richard jones* imperial college
scott phillips* texas a&m
jim downing* u cambridge
richard rodgers* mit libs
graham trigg* biomed central
Gabriela Mircea* toronto
henry jerez* cnri
rob tansley* google
Mark Diggory* mit libs

II. welcome/introduction

- dspace works
  - but dspace can't stand still
- libraries change
- libaries want to work with more stuff, higher volumes of stuff
- librarues want to work with other systems
  - courseware, etc
  - interoperation with various applications and systems
- people have had dspace for a while now
  - there is a legacy
- lots of development in the ingest process
  - BUT what about versioning, migrations
- HOw can dspace evolve to meet these needs?
  - "exec summary" or outline of where to go?
  - everyone is interested in dfiferent things, will pursue them
  - but here, develo outline

- dspace has been successful because core has set of basic features that work
  - not "research"
  - but people able to DO research
  - but people able to customize also
  - focus of DAR on core

- dspace must keep focus on keeping common tasks (of libraries) easy

III. review of materials

- issues list
- manifesto (http://wiki.dspace.org/index.php/ArchReviewWorkingPrinciples)
- other docs

- avoiding second system effect
    - that's the goal of manifesto

IV. Review of Manifesto

see: http://wiki.dspace.org/index.php/ArchReviewWorkingPrinciples

1. DSpace is primarily open source software for building digital repositories.
DSpace is intended to be free and open source software for digital repositories that enables services for access, provision, stewardship and re-use of digital assets with a focus on educational and research materials; i.e. to fulfill the mission of the DSpace Foundation.

(no discussion)

2. DSpace will be usable based purely on free and open source software.
Although setups including custom and/or proprietary features and technologies will be possible, it will always be possible to deploy DSpace using only free and open source software.

- issues with oracle
- issues with future java?

3. DSpace will have a decoupled, stable, and application-neutral core.
DSpace will attempt to identify a "core" of the system that supports a wide variety of applications, whose full scope is not bounded unnecessarily . It will define stable APIs to enable diverse and innovative applications and functionality to be built on this core, without need to modify the source code of the core.

- changed wording
- what is "core?" what is "out of the box?"
- four+ committers in room, not same def'n of "core"
    - out of the box?
    - some set of apis?
- core means something different re: future add-on architecture
- e.g. scott don't consider the appl part as part of the core
- rob: "core is what is needed for any reasonable application of dspace"
- jmo: need to define what set of services is in the core, and what a reasonable level of stability for the core
- md: strength of eclipse is that the "core" is well defined
    - also there are areas that are "priavte" or otherwise tagged
    - e.g. "this part will change"
- rob: currently, the apis and the application are currently THE SAME THING
    - need to objectify
    - reinforces the idea that there are places not to touch
- rob: a number of changes that have come in weren't necessary, but were done for expediency (they did it the "easiest" way)
- hj: core as LCD of the data model?
    - rob: there are extremes: "core" could be fs, core could be feature-rich application
    - jmo: historically its been sort-of the biz-logic layer
- sp: mental three-layer model
    - could define what is required of the application layer, call THAT the core
    - the lower you could, the more stable it will be
- sp: separate out the areas that will need to change
- should this group focus on application and storage layers?
    - md: current mit aips work
    - sp: if the md impl better, that work not necessary
- rob: back in the "dspace 2.0" discussions, thought of "core" as a set of apis
- jmo: agree on STABLE CORE, but what is that is ambiguous
    - separation of individual applications vs. what is "core"
- hj: focus on conceptual services offered by core, rather than the actual api
- CORE: "set of services and reference implementations thereof are what the dspace community should commit to remain stable"
- ms: dspace has been successful because it is an APPLICATION
    - hj: dspace as an application is confusing? is it an application, or a model, or???
    - ms: more like middleware, ie. that it is extensible
- ms: long-time tension: general purpose vs. doing one thing really well?
- rj: remember that this is a guiding principle
- TODO: Specify what is meant by "core"

4. Whilst usable for a variety of applications, DSpace will retain useful "out of the box" functionality for common use cases.
DSpace cannot support all the variable and emerging definitions and innovations in the repository space in a single interface application. DSpace will seek to provide out-of-the-box functionality for a common set of use cases (e.g. an OA preprints application, a general content archive) that can be installed with minimum possible effort, as well as modular support for the easy construction of new applications.

5. DSpace will employ and support existing, open standards where possible.
Wherever possible, DSpace will employ and support existing terminology, open standards and profiles. This is to promote interoperability of various kinds with other systems, and support the migration of data into or out of other systems.

- jse: where's the pain?
- rob: "lossy crosswalks," improper use of e.g. dc, etc
- rob: pay open standards due diligence
- rj: if we are adopting a standard, it should be open
    - it should be possible to exploit a standard, due to the openness of the system

6. DSpace releases should be minimally disruptive.
The architecture should reinforce good behavior in making changes/customizations/improvements to future releases of the system, so that upgrades are minimally disruptive for current adopters.

7. DSpace will support an exit strategy for content.
It will be possible to export all data necessary for the future re-use and stewardship of content held in a DSpace repository, in open and/or well-documented formats, for enabling migration into other systems and/or backup.

- rob: and/or...

8. DSpace will continue to evolve.
There are many unsolved problems associated with stewardship of digital materials, which will require research and experimentation (including some failures) to solve. In addition to providing a robust, stable and functional system, DSpace will enable innovation and experimentation, and will be designed with the knowledge that future development and re-architecting will inevitably be necessary.

BREAK

V. Review of Requirements

- rob: presentation of survey results

1. Use of DSpace

- total adds up to more than 100%
- REAL number is over 400...

2. Maturity

- mostly production or "pilot"

3. Version

- mostly 1.3 or 1.4

4. Use priorities/content in dspace

- lots of diversity
- LOTS of "other"

5. Metadata

6. updating

- majority have trouble customizing

7. Customization

- surprisingly high number implementing new features

8. Comfortabe customizing?

- survey sez majority confortable, but MS sez more like 50/50

9. Documentation

- majority fine with it
- lots of comments
- community likes to complain but don't contribute back
- e.g. they solve problems BUT don't contribute what their solution was
- educate community: they need to contribute
- get support

10. customization options?

- jse: several authz wants

11. Core code changes

-

12. User interface

- majority are doing lokk/feel

13. third party customizations

- srw/u (z39.50)

14. Handles

- majority use hdls

15. More important feature/function

- modularity + customizable ui
- versioning, complex objects, admin ui
- metadata
- workflows, rich media not so important

16. Main Problems

- many and varied

MK: talked to LoC

- dspace data model limited BUT
- dspace codebase "one of the cleanest they had ever seen"
  no*architectural obstacles

jmo: some at high-end want more from data model/application features

data model vs. ui?

hj: what is the canonical dspace use case?

- ms: wants to know that too...

What are the minimal changes to the data model that are the most useful

- small set, big impact???

What about scalability?

- lots of objects (volume)
- size of objects
- rob: already some things that can be done to scale up (e.g. operate on server cluster)
- browse with lots of items slow
- indexing on ingest/decoupling

Other server/infrastructure things that DSpace doesn't know what to do

SP: Can we put down size that we would like node of dspace to scale to?

- but what about other factor? What are the dimensions and the sizes?
- search engine calls, ai harvests

SP: Cambridge --** topping out with ~175,000 (many things breaking)
md/rob: questions of oai implementation

we need more well-designed tests to identify true bottlenecks

- e.g. "hp tr collection" (jimR)

Support for different "Activities"

- user vs
- admin vs
- question of arch support

dm: binding of permissions to items

interoperability

- info/dat vs api/protocol/service interop
- rob: srw good case-in-point
- rob: people can do whatever they want
- hj: mpeg21 import/export
  - also fedcor
  - ms: data interoperability

deposit (uk thing)

- policy issues
- rj: use case from imperial
  - pull data from authoritative repo
  - deal with licensing system
  - also pull workflow state

    - "lni" stuff ("lightweight network interface")
        - establishes "trusted relationship"
    - set up dual ds's, and three-step
- jd: point-to-point deposit scenarios

TODO: Wee need STATEMENTS about interoperability and scalability

-
    - and what is our scope

VI: Issues List
added issues:
jd: distribution/release management

- hj: probably need a set of tools/harness
- unit testing for dspace will
- md: use of maven in jakata community
  jd: communicating to community

ms: larger issue

- the stuff we're talking about requires infrastructure and resources
- We need to decide if this is the message that goes to the community and the DS steering body

jmo: What form of guidelines "published" to the community?

LUNCH

VII. Dealing with Issues

- RR: Should we do "triage?
    - i.e. prioritize?

jmo: having "strawman" proposals for each of these would be

See: http://wiki.dspace.org/index.php/ArchReviewIssues

1. Data Model/Information Architecture (Tuesday)

- versioning support
    - See VersioningProposal
- revisions in the identifier system
    - see e.g. this old persistent identifier discussion
- more flexible metadata options
    - often mentioned in connection with METS
- support for relationships between bitstreams
    - see BitstreamRelationships
- more robust content format support
- alignment with the JSR170 data model
    - see http://jcp.org/en/jsr/detail?id=170
    - interesting because:
        - looked at by a lot of people
        - specified interface with many possible implementations
        - leg-up over model with just GET/PUT
    - Q: why should DS export the interface?
- revisiting how aggregation is modeled
    - currently done via Communities and Collections
    - May dovetail with JSR170 approach
- Scalability?

2. Interfaces and Modularity (Tuesday)

- where to draw the lines...
- Rob to review current "apis"
- Big Issues
    - allow "wacky" innovations that don't require major changes
    - adopters are having trouble upgrading to new versions after having made changes
- RJ to discuss: The Add-On Mechanism
- What "shape" should a "framework" be?
- What about user interfaces?
    - JSP-based vs. Manakin
- Scalability?

4. Information Lifecycle management/workflow (Wednesday)

- Weds: ePerson/identity mgmt
- Authentication??
- Authorization/Policy stuff (john)

- AIP/SIP/DIP "stuff"
- Scalability?
- History and provenance?
- auditing, content authN and integrity
- Thursday Afternoon: Larry Stone

3. Concrete Model/Asset Repository (Thursday)

- storage-layer api?
  - related to jsr170
  - THIS is where the jsr170 discussion should be
- storing aips?
- two issues:
  - encoding issues (mets, mpeg didl, etc)
  - what to do with them
- Major discussion of implications of adopting something like jsr170
  - and therefore supporting third-party implementation of something behind an api
- Scalability?
  - Search and browse use case

BREAK

VIII: Scoping Goals

See also: Bakery of Half-baked Ideas

- Rob: These goals are qualified by this iteration

1. **Scalability:** DSpace should exhibit "reasonable performance" under the following conditions

- Items (static limits: "10 million items," "20 million unique indexed metadata values")
  - ingest performance
  - search/browse performance
  - implications: need test capabilities/test harness
- Bitstream size
  - no limit due to DSpace itself
  - practical limits due to http, file system: e.g. 2GB (http) via UI
  - goal: HD-DVD 50GB
- Dynamic limits:
  - generally: performance should degrade reasonably
  - batch ingest shouldn't take more than 1sec/item (???) up to static limit
  - ...and fully indexed in a day
- Users (concurrency)
  - 10 conc. updating users (response time?)
  - 100 conc. reading users (response time?)
- Bandwidth
  2. **Interoperability** DSpace should support...
- 3. **Release Management**
- JD: more modular approach?
- Rob: if that, then what about committer structure?
  4. **Core**