Planning for the next generation

A summary of recommendations for the next DSpace architecture

John Mark Ockerbloom
Open Repositories 2007
January 23, 2007
Why a new DSpace architecture?

• Use, scale, and dependence on DSpace growing
  – New applications continue to develop
  – Repositories growing older (with preservation needs growing too)
  – Patching, ad-hoc development only gets you so far (and may lead you into dead ends)

• Architectural needs
  – Set priorities for DSpace development, functionality
  – Handle the variety of content, metadata institutions manage
  – Make it easier to develop, customize, compose with other systems
  – No DSpace is an island

• Set directions for an evolutionary, practical system design
  – Serves community needs for several years, but take no more than a couple of years to produce
Architecture review group

- John Mark Ockerbloom, Penn (chair)
- Tim Di Lauro, Johns Hopkins
- Mark Diggory, MIT
- John Erickson, Hewlett Packard
- Jim Downing, Cambridge University
- Henry Jerez, CNRI

- Richard Jones, Imperial College
- Gabriela Mircea, University of Toronto
- Scott Phillips, Texas A&M University
- Richard Rodgers, MIT
- Mackenzie Smith, MIT
- Robert Tansley, Google
- Graham Triggs, Biomed Central
The process

- DSpace 2 discussions started in 2004
- In summer 2006, group chosen to review complete architecture
  - From DSpace committers, major developers, other stakeholders and architectural experts
- Online discussion
  - On Wiki, DSpace-devel and review group list
  - Manifesto, issues lists, survey
- Week-long “summit”
  - October 2006, Cambridge, MA
  - Came up with recommendations, proposals for DSpace 2
- Follow-on activity
  - Subgroups to address workflow, extension framework issues
  - Further development of data model, development roadmap
  - Report and presentation (You Are Here)
Survey

• Questions and comments about use and customization of DSpace repositories
  – Responses solicited on DSpace mailing lists
  – 116 responses in one week

• Adaptation common
  – Many customize metadata
  – About 1/4 change database schema
  – 1/2 made significant code changes
  – Problems keeping customizations, new versions in sync

• Commonly desired
  – Better modularity
  – More customizable UI
  – Complex objects
  – Versioning

• Full results, comments on DSpace Wiki
DSpace architecture manifesto: Part 1: DSpace nature

1. DSpace is primarily open source software for building digital repositories.
   -- Avoid scope creep (e.g. into general purpose CMS, Wiki…)

2. DSpace will be usable based purely on free and open source software.
   -- Avoid proprietary dependencies
   -- May still support closed source as option (e.g. Oracle)

3. DSpace will have a decoupled, stable, and application-neutral core,
   -- Not the full distribution
   -- Applications and extensions built on it

4. While usable for a variety of applications, DSpace will retain useful "out-of-the-box" functionality for common use cases.
   -- Standard distribution includes full open access archive
DSpace architecture manifesto: Part 2: DSpace development

5. DSpace will employ and support existing, open standards where possible and practical.
   -- Makes DSpace easier to develop, interoperate
   -- May make it easier to integrate other open source SW

6. DSpace releases should be minimally disruptive.
   -- Keep repositories stable
   -- Ease customization, maintenance

7. DSpace will support an exit strategy for content.
   -- “It’s the content, stupid.”

8. DSpace will continue to evolve.
   -- Because what DSpace users do and need to do evolves
Scalability

• Three scale dimensions of primary concern
  – Size of repository
  – Intensity of use
  – Rate of ingestion, other processing

• Group did not see major architectural limits to scale
  – but revisions need to accommodate large scale use

• Desired performance goals:
  – 10M items
  – 10 simultaneous depositors, 100 simultaneous users
  – 1 sec addition overhead at full size scale
  – Accommodation of clusters, unlimited size files
  – Ponies for everyone! (Okay, maybe not that…)
Interoperability

• Aspects of interoperability
  – Data interoperability (can I reuse, import, export info)
  – Service interoperability (protocols others can invoke)
  – API-level interoperability (extensions, new implementations)

• Needed for this:
  – Published concrete data model for content and metadata, fully exportable and importable
  – Published, documented, stable core interface
    » Designed with extensions in mind
  – Common, standard protocols supported in standard distribution
DSpace 2 highlights

- More powerful, flexible data model
- Shift in user interface model
- Core overhaul, documentation
  - to make extensions, customizations easier to add, maintain
- Focus on extended lifecycle of content
- More reuse of third-party development
  - Extension frameworks, workflow managers
Revised Item data model

Manifestations:
Replace Bundles, used for content only (some old Bundles would become metadata)

Content Files:
Replace Bitstreams

Can have multiple metadata records, attached to Items or sub-components
Identifiers

- Handles still default identifiers for content, but system should support others
- Persistent identifiers for Epeople
- Components within items should also have persistent identifiers
  - Proposal: URIs based on the Item identifier, with various qualifiers for Manifestation, Content File, and Version
Versioning

- Used for non-semantic revisions of content and metadata
  - Format migrations
  - Revised metadata
  - Possibly minor content corrections (typos, etc.)
- Semantic revisions (e.g. published vs. pre-print) can be separate items with Relation metadata to link
  - Not enforced by the system, but makes citation clearer
  - Will need metadata, UI support for ease of use
- Versions have their identifiers
  - No version specified = use latest version
- Retention of old versions matter of repository policy
  - But can be cheap to retain if not much changes
An example Item Version and its identifiers

![Diagram showing relationships between Item, Manifestation, Article Metadata, Deposit Licence, Content File, GIF, HTML, and File Metadata with identifiers X, X/v2, X/M2, X/v2/M2, X/M1, X/v1/M1, X/v2/M1, X/M1/C1, X/v1/M1/C1, X/v2/M1/C1.]

-derived from = X/v1/M1/C1
Further data model recommendations

• Metadata made more flexible, preservable
  – Managed and preserved in the persistent store
  – Multiple records supported
  – Serializable
  – Not constrained to be flat
  – Default schemas for Items, Content files…
  – Views of metadata can be projected into DB schemas for efficiency of access

• Separate abstract data model from concrete data storage

• Generalize Collections, Communities
  – But not yet recommending mixed-content Containers
User interface

- **We like Manakin**
  - XML-based interface makes it easier to customize, pipeline DSpace
- **Recommend adding it to DSpace 1 standard distribution**
- **Should become standard UI for DSpace 2**
- **Requires an add-on mechanism to integrate**
  - There’s a simple one now published for this purpose
  - But a more generalized approach could make it easier to add new DSpace applications, customizations...
Extension frameworks

• Extension or add-on mechanisms needed to integrate certain components (like Manakin)

• Reusing existing one preferred over doing one from scratch
  – There are a number of possible candidates (OSGi and Spring have come up as possibilities)
  – Requirements, discussion on Wiki
  – Implementation work, community input helpful in settling on one

• In the meantime, simple add-on mechanism released for handling Manakin and similar packages
Event mechanism

- Core should include event notification mechanism
  - Allows loosely coupled, open-ended components
  - Can be used to support history mechanism, view maintenance, UI
- How it works
  - Listeners register interest in certain types of events
  - Changes in data, other phenomena, raise events that notify appropriate listeners
- Prototype system developed under DSpace 1
  - Led by Larry Stone at MIT
- Details of DSpace 2 implementation may depend on decisions made for implementation frameworks
Workflow

• It’s not just for ingestion any more
• Again, can be supported by existing packages instead of rolling our own
  – Open WFE, OSWorkflow, jBPM identified as promising candidates
  – Other profiles, discussion on DSpace Wiki
• Better tools for specifying, modifying repository workflows needed in DSpace
  – Like user interfaces for non-programmers
The road to DSpace 2

- Core group (~3 people) does detailed specs of core, documentation, reimplementations
  - Not from scratch, but review needed of all core interfaces in light of new design
  - Goal: Have working DSpace 2 core within 2 years. Would not be responsible for entire standard distribution
- Architectural oversight committee (different from review group, but some overlap) monitors progress
- Wider community supports DSpace distribution effort
  - Developing extensions, applications, supporting and giving feedback to core group’s specs and docs
- DSpace 1 continues to evolve in the meantime
  - Manakin, Events, etc.
- Help build the next generation!
  - See full report, discussion on http://wiki.dspace.org/