July 5th 2010, OR10 Committer Meeting

Location

- UNED, Facultad de Humanidades

Agenda

10-1:30: Meeting
1:30-2:30: Lunch
2:30-6: Meeting

10-1:

- 3.4 release process, lessons learned
  - Agile process, milestones
  - timing of releases, are there certain times of year that are particularly bad or good
  - test plans for final release
  - people´s availability over the next month
  - Final release timing
- committer process
  - how meetings are working, getting the ideally twice yearly face-to-face
  - recruiting, fitting more people into the picture

2-6:

- longer-term roadmap (4.0 and beyond)
  - Revisiting London topic areas (for each: are we still thinking the same, what progress has been made, what's changed, how will we do them?)
    - high level storage
    - enhanced content models
    - Spring stuff
    - semantic web stuff
    - Drop box stuff
  - Topics that didn't make it to the London meeting
    - consolidating the APIs, authorization authentication
    - SWORD

Post meeting gathering:

- For the people at the University: you can walk (20 mins) through a nice park (althought it may be too hot at 18.00) or you can take the bus 46, right in front of the Facultad de Derecho, in the same street where is the building you are having your meeting. You end up in Moncloa. Take the street Princesa down, on your right (the same side where the bus left you) and walk to the next corner, after passing through and arch (from the bus stop only 1 min. walking). The bar is "Casa Manolo", and has good tapas and a lot of beer.

The url and the location map: http://11870.com/pro/manolo-1934-bar-restaurante

In any case, the area of Moncloa/Arguelles is full of bars.

Attendees

- Andrew Woods
- Aaron Birkland
- Brad McLean
- Kai Strnad
- Asger Blekinge
- Steve Bayliss
- Thorny Staples
- Bill Branan
- Chris Wilper (virtual Chris)
- Bill Parod (Northwestern U., USA)
- Louis Varita (UNED Univ in Spain)
- Matteo Boscini (CILEA, Italy)

Meeting notes

Morning session

Agile release management

1. Milestones were not "real" milestones - no consequences (unlike a software release)
2. Lack of available committer time was an issue for the release
3. Prioritisation and voting generally worked well, meant that non "pet project" items got worked on

Release timing

1. No overall best time for releases based on people's availability
    - Not a good idea to release right at Christmas or Open Repositories
2. Two releases a year roughly in October and April (similar to the Ubuntu release schedule) would make sense
3. More frequent releases than this are desireable, eg every 3 months
4. Aim for a 3 month release for 3.5, ie mid-November, plan at next committer meeting

3.5 Release

1. Bug fix release plus re-engineering of the release process (important to ensure we do have user-facing items in this release)
2. Two release manager roles for this release - traditional release managment and release process re-engineering management; Aaron may be willing to cover the former with Chris leading the latter
3. Release will include continued work on ECM and prep work for high-level storage (such as Spring integration)

3.4 Release

1. No change to the timing for this
2. Noteworthy outstanding items for this release:
    a. adapter for LLStore plugins (addressing Java package renaming)
    b. Datastreams: utility for migrating inline XML datastreams to managed content
    c. Datastream size: utility for calculating size of existing M datastreams (size now calculated automatically for new ones)
    d. Possibly new Fedora client (Asger)

Release process

1. Important to consider both what it takes for us to release, and what it takes for a user to install a release
2. Would be good to get closer to a "press the button" situation
3. Current process should be scripted
4. Consider doing nightly builds, this will help enhance the current release process by having a scripted process for generating the release artefacts
5. Nightly builds could also help engage the community more, for instance people who raise a bug/new feature (and perhaps provide a patch) would be able to test the implementation without having to do a source build
6. Do code coverage tests on nightly build
7. Clean out Maven repository as part of the nightly build
8. For users: upgrade process works reasonably well, but you have to "know what you are doing" (what directories Fedora uses and where these will be after an upgrade), this could be improved through better documentation and by including an "upgrade" option in the installer. A WAR-only deployment could also help this
9. Re-engineer release process in 3.5

Documentation

1. Currently, live documentation on the wiki is for the current release, previous version of live documentation is snapshotted to archive documentation for previous release
2. This is a barrier to documenting trunk changes, as it means modifying current release documentation (which then would not represent the current release)
3. Would be better to snapshot current release documentation as part of release process and link to that as the current release documentation (read-only, but allow comments, which would feed into next release), and have the live documentation reflect trun
4. There are still issues around this approach with changes done in branches (eg if branch doesn't make it into trunk for the release)
5. Atlassian have a space for each release, not clear if they have a "trunk"/dev version, we should look at their process in more detail to see if we can learn from it

Testing

1. System tests need enhancing
    - revisit what fits in config-A/B/Q test suites
    - do the current set of test suites still make sense?
    - should add testing for the various database, jdk, and OS variations that are usually part of release testing
2. Run code coverage tool (Atlassian Clover), do this overnight (as part of nightly build)
3. Have a JIRA issue for code coverage so this gets done at least once per release
4. Current test objects need revisitiing, need to step back and reconsider how the testing fits together

Getting more committers

1. Hold "special topics" meetings for "getting started with Fedora development" to lower the barrier to coding with Fedora in various dev environments (Eclipse, IntelliJ IDEA, Netbeans)
2. There is a need for review of patches, a "buddy" system could work, try and bring the person being "buddied" into committer meetings
3. Improve visibility of stuff being worked on, current priorities: publishing the "small items", announce on lists current priority of issues to encourage feeback/voting
4. Possible role for community management here, this could be the release manager, perhaps we should have a release manager who is not a committer

5. Report back to community outcome of committer meetings, particularly status of JIRA issues, JIRA issues opened/accepted; to dev and user lists

Roles for non-developers

1. Documentation gardener - keeps the wiki organized and up-to-date
2. Secretary - keep notes in all committer meetings and helps to keep the community up-to-date and involved

Face-to-face committer meetings

1. Open Repositories is the obvious choice for a meeting
2. Would be good to have the 2nd meeting of the year in the "other" continent to the Open Repositories meeting
3. Would be useful to have a wiki page noting up-coming events, conferences etc and who plans to attend; so we can better plan the 2nd yearly meeting

Afternoon session

High-level storage

1. Not a lot has changed since the London meeting (Data Conservancy work has affected availability)
2. Feedback needed in order to progress this (Aaron presenting at the conference, specifically requesting feedback)
3. Series of special topics meetings to be held to make decisions
4. There are big issues, need proper governance for these issues
5. Key themes in progressing this are a roadmap and governance
6. Development plans to be finalized at next committer face-to-face

Enhanced content models

1. Most of the ECM specification has been implemented for 3.4
2. The validate method has been implemented (outstanding work on RELS-INT validation)
3. Items on the roadmap:
    a. "Clone" method (3.5 release?)
    b. Search engine configuration for atomistic models
    c. Also noted that atomistic models are problematic in general for indexing, searching, OAI-PMH

Spring stuff

1. We held a special topics meeting to agree an approach (turn existing modules into beans)
2. Ben has a branch for the initial implementation (server becomes a bean factory); motivation for the branch was allowing trippi multiplexing
3. Next steps
    a. Fedora modules become beans
    b. Get rid of the Server class
    c. Could be done as part of the 3.5 release
4. Spring could potentially make future FeSL work easier
5. Spring would make JMS using Camel a lot easier
6. High-level storage has some depedency (though not necessarily) on Spring, post-3.5 release would be fine for this to support High-level storage

Semantic web stuff

1. Persistent http URIs for resources: Steve to document various options and have a special topics to decide, objective to agree this for 4.0 release implementation (as it may involve change to REST endpoints)
2. Issues around endpoints (/content for datastreams but not for disseminators), REST semantics (/content for disseminators would be wrong for POST-ing), using /profile and getting rid of /content, using /invoke on a disseminator, user expectations on endpoints
3. Named graphs: initial investigations on performance using views not encouraging (up to 10x performance degradation), more investigation to be done, potential alternative solutions to be investigated (eg "reference counting", storing "source" of triple outside resource index)
4. A tie-in between high-level storage and the resource index was noted, this might be a route for implementing different triple stores

Drop-box stuff

1. There is a basic working implementation (http://github.com/kstrnad/fedora-commons-dropbox)
2. This is a "folder watcher", FTP is also available

API consolidation, AuthN, AuthZ

1. We have two APIs (REST, SOAP) with two sets of code
2. SOAP splits into API-A and API-M, REST does not
3. Mapping SOAP directly to REST does cause issues, however we should perform a comparison and understand the differences
4. The Java client should map directly to the server API
5. Merging APIs should be targetted at the 4.0 release
6. Consider hooking authorization to objects/datastreams rather than API methods