TimeDrivenReleaseProposal

This proposal has now been accepted. The DSpace Developers moved towards Time-Driven Releases with 1.7.0 Release, and plan to continue that process for the foreseeable future.

Issue Addressed

Consider the time sequence of DSpace major releases (dates approximate):

- 1.2 (Feb 2005)
- 1.3 (August 2005) 6 months
- 1.4 (July 2006) 11 months
- 1.5 (March 2008) 18 months
- 1.6*(March 2010) 24 months *=projected

Despite a vastly larger developer community, more committers, better organizational support (DuraSpace),
and better software management tools and processes, our release cycles continue to lengthen. This not only creates the impression of a languishing project, but needlessly delays important new features and bug fixes to our adopter community.

Proposal

Move from a feature-driven to a date-driven release cycle. A conspicuous example for possible study is the Ubuntu Linux distribution, which releases twice annually, in the fall and spring. We may lack the bandwidth initially for such an ambitious schedule, but we could begin with maybe one annual major release, and allow interim smaller ones. The details are less important than the shift in methodology. Also interesting is Ubuntu's practice of designating every 4th release (i.e. every 2 years) a 'LTS' release, meaning long term support (i.e. they promise to patch and fix problems in these releases longer than the others).
This is a clever way of reducing maintenance costs, but still providing a stable platform for those who may lack the resources to upgrade to each release as it comes out (which describes many DSpace sites, I think).

To Consider

If done correctly, timed releases might actually encourage wider participation (developers, testers, etc), since there will be a known 'payback' time that is lacking today (I know my work will see the light of day in 3 months, rather than the indefinite future as is the case now). It might also improve the quality of contributed work, since it will reduce the temptation to push in insufficiently designed and tested work, out of fear that the next release is so far off.

Costs

There are costs to reckon with, however, which include an earlier 'lock-in' to participation in a release than our community has historically asked of it's members. That is, we will need to line up resources (release managers, testing dates, etc) much sooner than we have done.

Comments

What do you think?

Mwood 19:38, 10 March 2010 (UTC) A roughly periodic release schedule does force some planning to select a manageable amount of development within the interval. I think that's the real problem: we don't get the community together often enough to talk over what to do next, and then follow through. We get lots of ideas but we could do better prioritizing and scheduling them. We might benefit as well from earlier branching so that things which aren't ready for release.next always have somewhere to go.

Notes from Discussion on 10 March 2010

During the Special+Topics Meeting on 10 March 2010, we seemed to come to the following general consensus around this proposal:

1. We generally agree that Time-Driven Releases seem like a good thing
2. We generally agree that only major releases (1.6, 1.7, 1.8, etc) should be time-driven. Minor releases (1.6.1, 1.6.2, etc.) could just occur when bug fixes are necessary.
3. We generally agree that minor releases (1.6.1, 1.6.2, etc.) should *only* be for bug fixes. New features need to go into major releases.
4. We generally agree that if we went with a Time-Driven Release approach, we'd likely need to branch SVN more frequently. This would allow us to simultaneously work on two releases (e.g. a minor, 1.6.1, release alongside a major, 1.7.0, release) in two separate areas of SVN.

(*NOTE: If you feel any of the above statements misrepresent our discussion, please let me know! – TimDonohue*)

Questions left unanswered:

1. Mark Diggory and Graham Triggs would like us to also support asynchronous releases. For example, releasing just a 1.7.1 'dspace-api' module, if bugs only affect the 'dspace-api' (and not waiting for a full 1.7.1 release). How does this fit into time-driven releases? Could it be complimentary or does it require changes to this time-driven release idea?

2. If we were to approve this time-driven release idea, what sort of time-frame would we want to set? One major release per year? Two major releases per year?
    - For the time being, we've decided to schedule the 1.7.0 Release for Dec 2010. After 1.7.0 is released, we'll decide on an ongoing release schedule based on the experiences with releasing 1.7.0.