Thoughts on Version Numbering

During Developer Meetings on 2011-11-09 and 2011-11-16, we decided that the next DSpace Release in Fall 2012 will be called 3.0

Thoughts on Version Numbering

This page is a "placeholder" for all suggestions around renumbering our DSpace releases. Our current version numbering convention is documented in our Release Procedure at DSpace Numbering Convention

Overview - Why is this even being discussed?

In recent and past Developer Meetings, the topic always comes up: "I'd like to suggest we change our DSpace version numbering from the 1.x.x (e.g. 1.6.0, 1.6.1, 1.7.0, etc.) numbering we currently use." (See DSpace Numbering Convention, for more info on our current version numbering convention)

Usually this comes out of a few perceived issues with our current numbering sequence (whether or not these are just misperceptions, or are actual issues is up for constant debate):

1. All the "baggage" around the idea of DSpace 2.0, and wanting to avoid that in the future. The version "2.0" has been mentioned off and on for 5+ years as the "next, amazing version of DSpace". Therefore "DSpace 2.0" is sometimes perceived to have a lot of expectations around it (especially in terms of what features it must have to be considered "2.0").
2. The potential misperception that "1.x" versioning sounds like DSpace may not be as mature or stable as it really is. Obviously DSpace has been around since 2002, and has over 1000 institutions using it worldwide (which speak to the maturity and stability of the system). But, some may feel higher numbered releases sound as though they are more mature.

The Case for Date Based Version Numbers

So what do we mean by a date based version number? It's when the date of release forms part of the product identification - like Windows 95, or Windows 98. Or, more usefully in our case, Ubuntu's numbering scheme where the major number is the year, and the minor number the month. Which gave us: 10.04 (released April 2010), 10.10 (released Oct 2010) and the forthcoming 11.04 (to be released April 2010). For DSpace, it would make the next release 11.10.

Why should we adopt this scheme? Aside from the reasons given above (which are applicable to all alternative schemes), there are advantages to date based versioning:

1. It's a logical change - ie. you can see that the change is numbering scheme is clearly meant to reflect something specific, and not just an arbitrary chosen value.
2. Promotes the view of DSpace as an evolving platform (as opposed to the revolutionary / evolutionary cycles of standard major / minor release numbers)
3. It's clear how old your version of DSpace is without looking it up. In 2020, I'll be able to immediately tell you when DSpace 11.10 was released. How quickly can you answer that question for DSpace 1.4.1?
4. Reinforces the process of time-based releases, and favours a predictable schedule for subsequent (major) releases

The last point does also provide one possible disadvantage - if we ever wanted to back away from time-based releases, a date based version number would be impossible to pre-announce before it's clear when the release would take place (but we could always use codenames for development announcements).

Date Base Release Versioning with separate individual addon version numbering.

This would augment the above strategy with the ability to version individual modules (dspace-api, dspace-stats, dspace-discovery) with thier own version numbering (1.8.1, 2.0.0, N.N.N). There are further details on this here Asynchronous Release

The basic principle would be that the dspace/trunk/dspace directory would become the project that was versioned with the dated number (11.10) This may or may not ever reach a point of being placed into the Maven repository, as it is used as the assembly point for the application. *(On a tangent, actually creating artifacts for the configuration sources (config,etc,solr) would enable us to start to create Maven archetypes for starting a DSpace build rather than downloading source releases)*

Likewise, dspace-parent would go away and we would rely on dependencyManagement to be handled in the dspace/trunk/dspace/pom.xml such that the build/deployment has direct control over it at assembly time.. Using dspace-pom instead of parent allows us to release indivudal releases of dspace-api, dspace-xmlui, dspace-stats, etc When we then have the ability to release these separately, maintenance releases can simply be resolving the latest version of a maven artifact from the repository (via dependencyManagenet in the dspace/pom.xml) rather than having to actually upgrade the dspace /modules/*/pom.xml files to newer versions of the dependencies.

This actually turns our maintenance release process into more of an "update" to your currently configured build rather than a merging of sources. As long as customizations are maintained in dspace/modules and all that is required to update your release is incrementing the version number in your dspace /pom.xml dependencyManagement, then minor updates can be released much more frequently than 2-3 time a year, for instance, all the recent fixes on the dspace-1.7.x branch could have been release incrementally and actually be in effect on the DSpace 1.7 release already. This is why I think this will be such a powerful approach.

Please add your own proposals on version numbering here....

This seems to me like an ideal candidate for convergence with other members of the duraspace community. Differing to a wider policy both stops the conversation and is says a lot about the our future plans for the project as a whole. (stuart yeates)