Redmine - Feature #5391

progression of versions *per* *branch*

2010-04-27 20:48 - Daniel Miller

Status:	New	**Start date:**	2010-04-27
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Resolution:			

Description

In some schools of thought, major-release versions are branches that are never merged wholesale back to trunk or to any other release-branch. Then point-releases occur along all of several actively developed/maintained major releases. For example, observe GCC. Currently, gcc-4.3, gcc-4.4, and gcc-4.5 (and perhaps many others that are off my radar screen right now) are all actively and rather independently developed and maintained while ever-more brand-new work is happening on trunk. There can be many versions released along, say, gcc-4.5 for a while, during the same period that, say, relatively fewer versions are released along gcc-4.3. I am not focusing on a old GCC bug whose resolution is being merged or ported to all of gcc-3.3, gcc-4.1, gcc-4.2, gcc-4.3, gcc-4.4, gcc-4.5, and trunk. Instead, I am focusing on the release identifiers that bundle up multiple (different) resolutions along each of these independent branches.

Conversely, Redmine does not firmly have this concept of versions progressing along branches independently, concurrently. Instead, Redmine's current lack of capability implicitly implies a strictly linear set of versions. For example, roadmap is a linear nonbranching list, not an n-ary tree that could model what is going on with GCC as described above.

This is not only important for GCC in OSS. This is the way that industrial commercial software is often delivered to customers. Indeed, in defense contractors and telecom equipment manufacturers, it is customary to have not only a branch per major release, but rather: every contractual release to *each* customer is itself a "major release" that deserves its own branch with a linear progression of versions along that branch. In terms of GCC (used as a purely hypothetical example here instead of an arcade body of software on some hardware that you have never heard of), imagine that gcc-4.4 was delevered to the Navy, the Army, the Marines, AT&T, Verizon, and Deutsche Telekom. Not only would gcc-4.4 be an on-going branch (that is never merged wholesale with any other release-branch or trunk) that is separate from gcc-4.3 and gcc-4.5 as described above, no, the set of branches {gcc-4.4-Navy, gcc-4.4-Army, gcc-4.4-Marines, gcc-4.4-ATT, gcc-4.4-Verizon, gcc-4.4-DeutscheTelekom} would all be created. Why? Because the cost of certifying & deploying a release on to possibly 100,000s of machines can run into the millions of dollars per release. Thus, Navy wants to have its own gate-keeper to control how often new releases come in their door. Army wants their own gate-keeper. Marines, their own. AT&T, their own. Verizon, their own. Deutsche Telekom, their own. Along with all of this, each one wants their own report of what changed and precisely why, for what they call "traceability" so that, say, one telecom company is not precipitating troublesome changes into another telecom company's software, perhaps maliciously. This is why tools like Redmine (e.g., DevTrack, DDTS, ClearQuest) have existed at such companies for over 20 years as COTS tools (and even longer as in-house tools).

But in short, solve the complete GCC problem of concurrent-branch development & maintenance well and everything else will naturally fall into place for the even-more-elaborate scenarios too. Indeed, it seems that, analogous to the GCC situation, even Ruby would be needing this feature too for maintenance along the 1.8 version of the language that occurs concurrently with development & maintenance along the 1.9 version of the language.

History

#1 - 2011-01-07 18:24 - Andy Bolstridge

on the one hand, I feel that different long-term-support versions are managed as sub-projects. However, for small projects, a simple link to the repository would be more than sufficient to handle on-going support and updates to these parallel branches.