Redmine - Feature #35196
Version tracking (allow filtering out tickets which don't affect given version)

2021-05-02 23:25 - Philippe Cloutier

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Issues filter	**Estimated time:**	0.00 hour
Target version:			
Resolution:			

Description

I am new to Redmine, but one basic feature which appears to be missing is version tracking. There are 2 angles in which version tracking helps:

1. It clarifies which versions a given issue affects (mostly relevant to developers).
2. It allows to filter out issues which do not affect the version relevant (mostly relevant to users).

For me, #2 is the most important benefit. For example, Redmine currently has over 10 thousand issues reported. But it may be that fewer than 5 thousand of these affect the current version (4.2.1). It is therefore critical to properly evaluating a product to have the ability to limit displayed tickets. It is also seriously inefficient to search and report issues in mature software without such a feature.

A basic scenario is that if a certain issue x is solved in version 2.0, then that issue does not affect version 2.0. However, there are more complex cases. For example, that same issue does not affect version 2.1, as version 2.1 is ulterior to 2.0.

Unfortunately, this can get tricky. There are cases where a solution is reverted. For example, if issue A is reintroduced in version 2.2, then version 2.3 may be affected, even if it is ulterior to 2.0.

Even for simpler cases when an issue is solved once for good, branches complicate things. For example, if issue B affects version 1.0.0 but is fixed in version 1.0.3, is version 1.1 affected? Without more information, that cannot be answered.

The ITS engine which properly implements this that I know is Debbugs: https://lwn.net/Articles/144106/
A demonstration of aspect #1 can be seen on this libkate issue's graphic
Red versions are affected, green versions are not. Grey versions are those about which the ITS cannot decide.

Debbugs's approach is to scan changelogs to compute a version ancestry tree. This is very efficient, but based on Debian's mandatory and uniform changelogs. This approach won't readily be usable with Redmine. An alternative would be to enhance versions management allowing administrators to manually build their product's version ancestry tree, or to start with basic semantic inference (if an issue is solved in version 2.0.0, it does not affect versions 2.1.0 and ulterior).
There are several types of version indications possible.

1. The best indications are certain, based on code analysis. For example, if a commit which clearly introduces a defect is made between versions 2.0.1 and 2.0.2, then we can indicate in the ticket that version 2.0.2 introduces the issue.
2. Very often, indications are unfortunately not so certain, and based on behavior. For example, if a test reveals a defect in version 2.0.1 and the same test does not reveal the same defect in version 2.0.0, we have an indication that version 2.0.1 may have introduced that defect, whose certitude depends on how reproducible the bug is.

Ideally, tickets could take precise indications about issue introductions and solutions, and Redmine would be able to tell, for any given issue and product version, the risks that it is affected. For example, a given version may:

- **Not be affected** by issue *x*
- **Be affected** by issue *y*
- **Have a 1 in 5 chance of being affected** by issue *z*

In practice, that would be quite an achievement. It would already be vastly superior to match Debbugs and simply manage to tell **whether an issue affects, doesn't affect, or may affect** a version, and I will consider this issue solved once that level has been reached.

Building blocks necessary to achieve this include the aforementioned version indications (possibility to mark certain versions as affected and/or introducing issues, as well as the converse possibility to mark as unaffected and/or solving). This building blocks are treated in ticket #685 (found in version) and ticket #219 (fixed in multiple versions). Defining a version inheritance tree is related to ticket #18126.