redmineorg-copy202205 - Vote #73509

Improving Redmine's version model (not just milestones)

2022/05/09 16:45 - Admin Redmine

ステータス:	New	**開始日:**	2022/05/09
優先度:	通常	**期日:**	
担当者:		**進捗率:**	0%
カテゴリ:	Roadmap_22	**予定工数:**	0.00時間
対象バージョン:		**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/13387	**status_id:**	1
category_id:	22	**tracker_id:**	2
version_id:	0	**plus1:**	2
issue_org_id:	13387	**affected_version:**	
author_id:	4	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	28		

説明

I've done extensive reading on past conversations about Redmine's version model. I'm not impressed by the suggested changes, which are kludgy and narrowly focused on strict definitions of terminology (milestones, phases, sprints, versions). The use cases are still important, so I hope this addresses most of them.

TL;DR
Evolve the version model so that it approaches the issue model, but retaining the cross-project visibility (sharing) system.

Detail
Most enhancements to versions are functionally one of two things: tags or hierarchical. We either assign values to versions for the sake of filtering them later (tags), or apply a structure to versions so that there are parents and children (hierarchical). Neither of these address all use cases, and both have been considered models for implementing milestones alongside versions.

I propose a three step solution.

1. *Rename*. Just as there's no need for a dozen trackers with the same fields and workflows, there's no need for half a dozen names for the same thing: dated objects work can be measured against. Versions, milestones, sprints, and releases are all Deliverables. Let's call them that. All of them.

2. *Evolve the model*. Deliverables need more versatility, and the issue model provides it, starting with relationships. Let's face it, a great deal of use cases could map deliverables with a Gantt chart, dependencies and all. Relationships allow it:

- Follows/precedes. Let a deliverable have a delivery date and/or a duration. If v1.1 comes 30 days after v1.0 and v1.0 is delayed a week, v1.1 follows suit.
- Parent/child. Let this solve the milestone-version debate once and for all. By keeping the existing version sharing system, parent/child deliverables can be tracked in the simplest and most complex project structures. Using milestones to coordinate many different component/subproject versions? Set up parent deliverables in the root and relate them to child versions in the subprojects. Using milestones to track phases within a version? Set them up in the same project as children. Gold. Filtering to a deliverable with children will list all issues associated with it and them, for all-inclusive progress reports if desired.
- Duplicates. Too late to resolve poor planning and version configuration? Just want to quickly build a report without updating 200+ bugs' target version? Make v.1.0 a duplicate of v1.0 and your v1.0 report will include all issues targeting the mistake.

1. Lastly, and where the functionality comes to life, *a filter UI*must be added to the Roadmap. This lets you build your query around exactly what deliverable data you need, and will immediately make all of those version custom fields more powerful.

I believe implementing this would require two changes to the database: (1) expanding the existing versions table and (2) establishing a version_relations table. Afterwards, the version create/edit form would need to be updated to write these values, version drop-down fields for issues may want to indent parent/children if necessary, and filters would need to be added to the Roadmap.

I'm fairly confident this model addresses ALL past requests for version improvements, but I'm always happy to be proven wrong!

journals

I've added ticket relations to all those issues you're added this information. If I've missed one, please let me know.

Thanks. Aside from issues, this also relates somewhat to the discussion over at message#214. That's just for reference, though.

I had some extra thoughts on the topic after posting, so I'll record them here--

-One enhancement would be adding a check box to 'version' configuration (adding or editing) to enable/disable letting issues target it. I was thinking about my many-subproject-versions-with-master-project-milestones example and that sometimes a milestone is not meant to have issues associated with it. While this could be managed with good discipline and review cycles, the option seems unobtrusive where it isn't needed and helpful where it is. A default value of Can be targeted would make the most sense, since it will be the vast majority of scenarios (I think).- *NOPE* -> I just realized that versions already have a 'Locked' state which accomplishes exactly this. Nevermind! 8)

I can also understand if the term 'Deliverable' isn't always satisfactory. Truth is, we're already using a feature called a 'road map', so road-friendly terminology might be better suited. I don't know. I put most of my thought into the model and just wanted a universal term for what we called the objects. It's hard to find that perfect name, and this is evident on all kinds of planning tools (not every Redmine 'issue' is an issue, not every MS Project task is a task, etc). Maybe we should just call them *Targets*, and change the issue field from 'Target Version:' to simply 'Target:'. Heck, we could just gamify the whole thing and call them 'Achievements'!

Also, a change like this is very baby-step friendly. Building a relationship model (the hard part?) could be tackled first, and everything else treated as future enhancements since they're mostly about interacting with the model. With relationships out of the way, the road map would be 'good enough' enough without filters (since we're re-purposing 'versions', everything created under the new model would already show up in the side bar), as long as the relationships are respected in the results (displaying issues for duplicate and child versions).

...and yes I was kidding about calling them Achievements. Calm down.

--

Actually, you know what? I like Targets now. Can you get any more generic for an object that issues...er, target? I may be missing some problems that would arise, but solutions can't be that hard. Have an 'Affected Version' field? Of course you do. You're a good tester. Leave it alone, or just call it 'Affects'. Yup. I like it.

I did start some work on this:

https://bitbucket.org/StrangeWill/version_details/

However, I strongly agree that Versions need to be first-class citizens like tickets, and have comments, change history, and all that good stuff. Getting versions to support a lot of the stuff Issues supports takes a very little amount of work, and I've been tempted to work on mainlining this instead of making this a mod.

Joshua - I'm stunned. Thank you for putting so much thought into this!

I'm fairly new here and don't know the reasons why the version model evolved like it did, but I can follow your thoughts and think your conclusions are true.

@William - I've seen your project but haven't had a chance to look through it yet. I'll take a peek when I have some time! I like your idea of change history for versions. They already support linking to wiki pages, which is actually pretty snazzy and I like what it can offer, but as actual values change, a visible history would be nice. The version table is already recording the 'last update' value, it just isn't tying that time stamp to a journal. I'd put it in my 'enhancements' bucket I described above, since it isn't a fundamental part of the new model. Definitely a great goal, though.

@Jan - Thanks for the positive feedback! My primary goal here is expanding functionality without introducing intimidating complexity (hence the familiarity of issue relations), maintaining upgradability (a RM improvement vs. a plugin), and establishing a system that

can adapt to a wide array of use cases without getting caught up in semantics. Redmine is in a great position to evolve without requiring disruptive changes, and that's a big positive. At least for the version model, everyone can get what they want simply by carefully expanding functionality in just the right spots. Redmine devs have done a phenomenal job avoiding 'too late to fix' traps, and I'm always impressed by what's possible for the future of this tool.

Good to see there is still people interested in this topic.

I think that to get a better, or more global view of the meaning of things, and to obtain the most versatile implementation in RM, one should quit the world of software development for a moment, and think he's building bridges, writing novellas, or any non-software stuff. Then suddenly Version takes much less sense.

Milestones are points of interest on a roadmap.

Targets are Milestones, but not all Milestones are Targets.
Achievements are Milestones, but not all Milestones are Achievements...
(and rarely are Targets and Achievements on the same points in time ;-)

Trying to give another "more accurate" name to milestone would just restrict its meaning and then its usability.

If I was to program a Milestone base class, I would just give it a name and a date for properties.

Joshua DeClercq wrote:

> @William - I've seen your project but haven't had a chance to look through it yet. I'll take a peek when I have some time! I like your idea of change history for versions. They already support linking to wiki pages, which is actually pretty snazzy and I like what it can offer, but as actual values change, a visible history would be nice. The version table is already recording the 'last update' value, it just isn't tying that time stamp to a journal. I'd put it in my 'enhancements' bucket I described above, since it isn't a fundamental part of the new model. Definitely a great goal, though.

It's very rudimentary and has some nasty bugs stability wise, I spent a couple hours on it for work and never got around to fixing it up...

We needed the ability to have a set of comments on a version, so we can discuss things such a overall status, release dates/procedures, etc.

@Eric Voisard
I disagree that usability would be affected. You're talking about terminology (or semantics, which I described with no lack of disdain earlier) at this point. Features and functionality would not change simply because the name is different.

From a purely functional perspective, we are assigning issues/tasks to these dated objects. We are <u>targeting</u> them for closing issues. Even if you called them Milestones, Redmine's issue form would literally read: "Target Milestone". The same can be achieved with 'Target' simply by naming an individual object 'Milestone x'.

¦My model¦Other model¦
¦Target: [Milestone X1]¦Target Milestone: [X1]¦

No matter which way you slice it, Redmine is already using the word <u>target</u>. It's already translating the word <u>target</u>. Whether we call them versions, milestones, phases, or cupcakes, the word <u>target</u> will be there. It's all about whether the next word is in the field's label, or the object's name. We can put it in the field's label and these debates will <u>*never end*</u>, or we can put it in the object's name, custom field, or anywhere else, and everyone gets what they want however they want it.

@William - For the sake of having it on the record here, I'd also like to observe that your comments system should also take advantage of email notifications and the Activity feed.

Well I think the basic idea is good.

A Version could consist of a variable amount of Milestones.
Some parts are not Versions just Milestones (for example you build a home, there is no Version 1.0 of your home ;-)).

Some more variable Roadmaps would be great.

These are mine Points of interest for a good Roadmap module:

- a Roadmap must be hierarchical (Version 2.3 consists of Version 2.3 - core and Version 2.3 plugins, for example). Even a combination of Milestone and Versions should be given. (Milestone is parent of Version or Version is parent of Milestone, even Version is parent of Version or Milestone parent of Milestone).
- a Roadmap should contain some type of wiki (currently given) and a even should be commentable. If there is some way of discussion.
- a Roadmap should be linkable. For example: Link a News item to a Roadmap and vice versa. Link issues to a Roadmap (currently given)

Maybe these Points are interesting for you too.

Best regards,

Daniel

Daniel Felix wrote:

> - a Roadmap must be hierarchical (Version 2.3 consists of Version 2.3 - core and Version 2.3 plugins, for example). Even a combination of Milestone and Versions should be given. (Milestone is parent of Version or Version is parent of Milestone, even Version is parent of Version or Milestone parent of Milestone).

I believe this could be achieved fully using the relationship model. If both 'versions' share a common parent, the road map for that parent would include a break down of each child version--in your case, components.

> - a Roadmap should contain some type of wiki (currently given) and a even should be commentable. If there is some way of discussion.

Agreed. I'm currently thinking about whether it's better to attach comments directly to the version or go for a commentable wiki approach. Versions already allow for wikis, and there are <u>many</u> scenarios where wiki comments make sense. Offloading the discussion to these pages offers the user more ways to access them (via road map or via wiki) and might provide a better value. But I'm still thinking on it.

It might be better to broaden what a version can reference: wiki page, forum thread, or news post. That raises another question of whether the user should specify just one, or all/as many as they'd like. But then comes the question of which one is fed to the road map view, and maybe that selection needs to be offered, too. Much further than that, and we're getting dangerously close to Overwhelm land.

> - a Roadmap should be linkable. For example: Link a News item to a Roadmap and vice versa. Link issues to a Roadmap (currently given)

This is technically already possible using wiki syntax version#ID or version:"name". With proper relationship configuration between versions, linking to common master at any preferred level would let the existing syntax achieve what you want without requiring any new code.

Unless by "linkable" you mean more like a relationship--an actual field that can be set to a point on the road map. I'd have to think about that, but I wonder if what I wrote under your second point might be close to what you're talking about (referencing news posts instead of just wiki pages).

Joshua DeClercq wrote:

> Agreed. I'm currently thinking about whether it's better to attach comments directly to the version or go for a commentable wiki approach.

Yes this could be some idea. Maybe just add comments to wiki or add some Kind of Forum thread to each wiki page which could be optionally displayed.

Joshua DeClercq wrote: ´

> Much further than that, and we're getting dangerously close to Overwhelm land.

Yes I totally agree. But if those Information are hidden and just be displayed if they are requested (something like my idea of tabbed journals).
This could give the user the abbility to get all the information of interest and still have a small page at start.

> This is technically already possible using wiki syntax version#ID or version:"name". With proper relationship configuration between versions, linking to common master at any preferred level would let the existing syntax achieve what you want without requiring any new code.

No this is not what I've meaned. This is pretty clear to me. :-)

> Unless by "linkable" you mean more like a relationship--an actual field that can be set to a point on the road map. I'd have to think about that, but I wonder if what I wrote under your second point might be close to what you're talking about (referencing news posts instead of just wiki pages).

This is more likely what I've meaned. Something like the ticket Relations in redmine. Just to have a link box to see what issues (!), news, comments, revisions (!) and maybe users work on which version/milestone (this could be handled by selecting all asssigned users of all issues in this milestone/version)

@Daniel
I like where you're coming from, but some of your ideas terrify me in terms of scope. :P But there's always a simple path. It just needs figuring out.

I had a big ol' post written up where I was working out a bunch of your ideas into a model, and it turned into a big, chaotic mess. The simple path is eluding me for now, but that's probably because I'm not in the shower. I can't solve problems without a shower. :P

I found this issue when looking around to see if there were any plans for a hierarchical versioning system. I notice there hasn't been much activity since the initial flurry -- I hope this is still an active issue.

I would strongly favor a single object type (eg. "target" as described at #13387-8) over a more rigid ontology. Just in our group, some projects would like to use version/milestone, and others use release/sprint. Also if you have more complex relationships like follows/precedes, I can envision things like parentage as an ad-hoc grouping mechanism -- say, targets X, Y, and Z (and perhaps others TBD) are all gated by targets A, B, and C, I might want to create a parent A/B/C target, and a parent X/Y/Z target, and add the follows/precedes relationship there. I wouldn't want to be constrained by (or even care about, for that matter) the "level" of a particular target in the hierarchy.

I'd also add that I would consider a simple hierarchy to have a much higher return on effort than some of the other features. I like the idea of casting it as a relationship, such that additional relationships might be added in the future, but I think interactions among relationship types (like the scenario above) could be devilishly complex, so tackling all of that at once seems liable to bog down severely.

Like Adam, I stumbled upon this issue while looking for hierarchical/nested versions.

Right now, we use Projects to simulate main versions and Target Version to manage each iteration. Even if it works, it feels awkward and the big picture is difficult to get. Can we have some feedback on that ?

A simple parent-child (Milestone-Version) hierarchy would be very nice and welcome functional change.

Note that this can already be partially achieved by creating a *Version* custom field ("Milestone") of type @version@.

However, the Roadmap view doesn't have the logic to display the roll-up.

I'd love to get possibilities to use at least 2 level hierarchy for versions. Any chance to get this soon?

+1 to many of the comments in here.

The overall version model should improve not only from the Milestone/Version/target as hierarchy - but also in terms of it being comprehensive structure.

Add related: #23285 #23286 #23287 issues which speaks other aspects to enhance the version model.

+1 Version to have history (tracking field modification, linked issues, etc).

related_issues

relates,New,374,Support for milestones/iterations as part of projects
relates,New,8991,Nested versions for projects with sub-projects
relates,New,724,change "versions" to "milestones"
relates,New,18126,Allow setting up version hierarchy
relates,New,17907,Give 'version' another meaning
relates,New,23285,Version to have history
relates,New,23286,Customizable state model workflow for versions
relates,New,23287,Better structuring of Version page
duplicates,Closed,453,a version includes many versions

履歴

#1 - 2022/05/10 17:13 - Admin Redmine

- カテゴリ を Roadmap_22 にセット