ChiliProject - Feature # 971: Multi-Repositories for a single project

Status:	Open	**Priority:**	Normal
Author:	Daniele Segato	**Category:**	SCM
Created:	2012-04-08	**Assignee:**	Felix Schäfer
Updated:	2013-01-08	**Due date:**	
Remote issue URL:			
Affected version:			

Description:

Hi request a new feature: being able to associate multiple repositories to a single project.

It's a good practice to split your project in multiple module and, sometimes, its a good idea to keep them in different repositories.

The only way, currently, to associate multiple modules to a project is to create subprojects.

But then you may have a bug "crossing" between modules and there's no way to create subissues / share that issue between subprojects (see #855)

History

2012-04-09 03:58 pm - Manuel Pais

Just to add at my company we are also facing the need to have multiple repos associated to the same project.

We use Chili (and Mercurial) on a project basis but we also have a large shared codebase on a separate Mercurial repo.

Often we find issues that turn out to require a fix in the shared codebase but because the SCM integration only checks the project repo the fixed shared code don't get referenced in the issue.

Thanks.

2012-04-15 07:01 am - Gabriel Mazetto

There is already a Pull request for this one : https://github.com/chiliproject/chiliproject/pull/179

2012-04-15 07:03 am - Gabriel Mazetto

This is related with this remote issue: http://www.redmine.org/issues/779

2012-06-16 01:35 pm - Gabriel Mazetto

- Target version set to 3.3.0

I'm assigning it to 3.3.0 as a way to put it on discussion. Also I want to point out: https://www.chiliproject.org/boards/2/topics/455.

2012-06-16 02:24 pm - Daniele Segato

Hi Gabriel,

thanks, I've read the forum post you linked.

So you don't see an use case for this.
I think I outlined one in opening this feature request, why don't you see it as an use case? How would you handle that situation without multi-repository per-project?

Can I also ask for a list of difficulties involved with evolving chiliproject to a multi-repo per project architecture?
Maybe, even without actually developing, we can be helpful to the topic by discussing it more in deep.

Do you prefer moving the discussion to the forum post you linked?

as a side note (probably a little OT): you talked about "top priorities" for chiliproject, where can I read the list of priorities?
how do you validate those priorities with the community?

Thanks

2012-06-16 05:33 pm - Felix Schäfer
- Assignee set to Felix Schäfer

- Target version deleted (3.3.0)

Daniele Segato wrote:
> So you don't see an use case for this.
> I think I outlined one in opening this feature request, why don't you see it as an use case? How would you handle that situation without multi-repository per-project?

The OP seems to say "we'd need multiple repos per project or the ability to have better cross-project issues". Feel free to correct me if I'm wrong.

> Can I also ask for a list of difficulties involved with evolving chiliproject to a multi-repo per project architecture?
> Maybe, even without actually developing, we can be helpful to the topic by discussing it more in deep.

Discussing it would probably be better than not doing anything at all, sorry for that.

Regarding the code provided, I'm not sure if anyone has reviewed it yet, I'll try to see to it in the next few days.

As to the "difficulties", I think it boils down to:
* Is the added complexity (Code and UX) worth it for the majority of users?
* How does multiple repositories per project fare with regards to the current "idea" is/should be?
* Could this be achieved another way (make a parent repo with all the repos as submodules/whatever svn calls them and update the SCM adapters to checkout those?)
* Is multiple repositories per project the best solution (wouldn't improvements to categories/milestones/issue sharing solve this issue too?)?

So to further the discussion (and I'm sorry if someone's answered this already, I'd appreciate a link to the answer :-)): What are the biggest problems with multi-module software projects? From what I've read: having shared issues or at least issue inheritance cross projects or at least their hierarchy, and commit messages only noticing relations to issues of the project the repository is associated to. What other problems are there?

> Do you prefer moving the discussion to the forum post you linked?

I think I prefer having it in the issue when there is a ticket, so let's stay here.

> as a side note (probably a little OT): you talked about "top priorities" for chiliproject, where can I read the list of priorities?

The next "big" priority is to move to Rails 3 (probably latest), the first half of this year has been pretty crazy for all the core members, so we're a little behind on the outside communication and documentation sides of things, I'm very sorry for that (see also the last paragraph of the "3.2.0 announcement":http://blog.chiliproject.org/releases/chiliproject-3-2-0-released/).

> how do you validate those priorities with the community?

Well, the current state was rather that we (the core contributors) haven't seen "much" community at all, so the decision was rather a core member committing to something he was interested in/which bothered him most. In addition to that we had semi-regular meetings on IRC where everyone could participate in.

We're seeing more involvement happening though, for example Andrew's work on css/js matters, and people "complaining" (which is a good thing, complaining is also caring :-)), so we need to work out some ways to further the discussion and involvement. If you have ideas on how that can be done and/or experience with other OSS projects (it's for many of us the first project we're "leading", for me it's even the second project I'm involved in in any significant way), we'd be more than happy to hear about them, but they should probably really go in a new thread in the forums :-)

I hope I was able to answer you well enough, if there's more you'd like to discuss you can open forum threads or catch me/us on IRC or Twitter :-)

2012-06-16 08:03 pm - Gabriel Mazetto
Thanks for answering this topic Felix, I will give a try to the pull request and try to give some feedback to it, as a way to contribute with this issue. I also would like to suggest that we add a version that could help plan things for future releases like "next major version" or something like that, as a way to prÃ©-select tickets

2012-06-17 11:57 am - Daniele Segato
Felix SchÃ¤fer wrote:
> The OP seems to say "we'd need multiple repos per project or the ability to have better cross-project issues". Feel free to correct me if I'm wrong.

well.. that's the "big" part of the issue, with the managing of user roles for cross-projects.

nevertheless I think having the ability of attaching more then one repo to a single problem is a more elegant way sometimes, and it can be "in addition to" the better cross-projects issues.

I give an example:
A git project with git sub-modules, having the ability to include the main project repository AND the sub modules repository to the same project may be useful.

> Regarding the code provided, I'm not sure if anyone has reviewed it yet, I'll try to see to it in the next few days.

I didn't looked at it.. without knowledge of the chiliproject source code I would think the most hardest part to get right in introducing multi-repo per project is the resolution of the repositories links in the wikis

> As to the "difficulties", I think it boils down to:
> * Is the added complexity (Code and UX) worth it for the majority of users?

how can you reply that question?
issue voting?
polls?
that can be an argument for the next IRC meeting, maybe?

don't know :) that's why I was asking how you validate the roadmap with users / community.

> * How does multiple repositories per project fare with regards to the current "idea" is/should be?

Probably the best way to get this right would be to define repositories "system wide" and associate them to projects, so that the same repository can be shared between project (in the submodule example this is right)

It would probably also simplify other things in the long run.

> * Could this be achieved another way (make a parent repo with all the repos as submodules/whatever svn calls them and update the SCM adapters to checkout those?)

as I said, I think you should take out the need of a project associated to a repo to make things simplier :)

as it's now we are creating multiple projects, setting parents...
but than it's hard to manage release roadmap, issues and all sort of things
and there's also a parent project that's empty

> * Is multiple repositories per project the best solution (wouldn't improvements to categories/milestones/issue sharing solve this issue too?)?

I think they are two different thing, they both can help, the issue/milestones/categories probably more then multi-project, but I don't think they are exlusive.

> So to further the discussion (and I'm sorry if someone's answered this already, I'd appreciate a link to the answer :-)): What are the biggest problems with multi-module software projects? From what I've read: having shared issues or at least issue inheritance cross projects or at least their hierarchy, and commit messages only noticing relations to issues of the project the repository is associated to. What other problems are there?

It's hard to reply, because it's a complex matter.

Something I can think about on the fly:
* parent task can't be cross-project nor refer to a parent project
* referring to repository commits in other repository from an issue is not very user friendly :)
* it's hard to visualize the big scheme of a project splitted to multiple projects / repositories [1]

[1] I see the need of having a roadmap be splitted into small roadmap achievements, like "roadmap parent" and "roadmap childs"
developing mobile application we found often with at least 3 projects (more sometimes): the Android project, the iOS project, the Services project;
having a way to define a roadmap to 3.0 (project wide) that include
* an android roadmap for 2.4.0, 2.4.1, 2.5.0, ...,
* an iOS roadmap for 2.1.0, 2.2.0
* and a backend roadmap 2.9.1
to be part of the "big roadmap 3.0) would be nice

This is a small list, I just tried to start some discussion about the matter.

> The next "big" priority is to move to Rails 3 (probably latest), the first half of this year has been pretty crazy for all the core members, so we're a little behind on the outside communication and documentation sides of things, I'm very sorry for that (see also the last paragraph of the "3.2.0 announcement":http://blog.chiliproject.org/releases/chiliproject-3-2-0-released/).

I understand your desire of moving to a newer technology, it makes the development smoother, its interesting to keep up to date with new features, but think about your question: "Is the migration to a new technology worth it for the majority of users?"

Be careful, it's very easy to end up moving from a cool technology to another and forget about users, which require new features, more ease of use etc... I understand it very well, I'm a developer too, marketing is there to makes me remember you "sells" features to users, not cool underling architectures :P
It's a good think you keep up with technology but do not focus on that too much, this is just my humble suggestion.

> > how do you validate those priorities with the community?
>
> Well, the current state was rather that we (the core contributors) haven't seen "much" community at all, so the decision was rather a core member committing to something he was interested in/which bothered him most. In addition to that we had semi-regular meetings on IRC where everyone could participate in.

I'm talking as an outsider here.
I not skilled with ruby and have a very limited time BUT, if let's suppose I wanted to start being part of the community: where can I found the information you have regular meeting on IRC? when / where are them? where can I read about this information?
Do you keep a log of the IRC meeting or an abstract of what resulted from that meeting somewhere?
Where can I read a _vision_ of the project that tells me the long-run direction you want to take?

In the roadmap there are currently two versions: 3.3.0 and 4.0.0. The first one has one feature, the second has 2 features. No description for both of them.
What's the target for those roadmap?

I try to give an example, if you write:
4.0.0: In this version we will concentrate on permissions, we will add comments visibility etc....
Then the issue list somewhat "describe" what's needed to get there.

Another way can be place features request from user and stick them into a roadmap, trying to keep them related

So that an user can see where you want to go and, if he need that can contribute thus becoming part of the community.
I have difficult seen where this project is going.

You did lot of work in the UI recently and I see you've done a good job.
But in the mean time you broke compatibility with many (redmine) plugins: thus removing features for some user.

> We're seeing more involvement happening though, for example Andrew's work on css/js matters, and people "complaining" (which is a good thing, complaining is also caring :-)), so we need to work out some ways to further the discussion and involvement. If you have ideas on how that can be done and/or experience with other OSS projects (it's for many of us the first project we're "leading", for me it's even the second project I'm involved in in any significant way), we'd be more than happy to hear about them, but they should probably really go in a new thread in the forums :-)

I would write roadmap description like in "intent" to bring user features.
In the overview explain you are under a technological review to makes introduction of new features easier and faster but THEN explain which new features you are going to introduce.

Ask user to tell you which features they would like to see, if you ask me I would say:
* better git integrations
* better cross-project relationships
* visibility permissions
* better time-management / tracking features
* plugins backporting / backward compatibility layer (even documenting the changes that brake compatibility, by versions, would be a lot useful).
* RESTfull services ease-up integration with external tools (jenkins, artifactory, ...)
* documentation (plugin documentation is almost empty)

Of course other users may have a different direction and it's up to you, in the end, to chose one, the important thing is that you choose one for the short period, medium period and long period :)

> I hope I was able to answer you well enough, if there's more you'd like to discuss you can open forum threads or catch me/us on IRC or Twitter :-)

Thank you very much for your reply.
I'll try to participate, as spectator, to the next meeting on IRC if I'm able too.

2012-07-10 11:20 am - Felix Schäfer
Damn, it took me longer to come back to this :-(I'll change the order of my answers a little bit, but I think the quotes should be good enough for everyone to be able to follow the whole thing :-) I think I'll split the answers too, those long posts are hard to followâ€¦

> > The next "big" priority is to move to Rails 3 (probably latest), the first half of this year has been pretty crazy for all the core members, so we're a little behind on the outside communication and documentation sides of things, I'm very sorry for that (see also the last paragraph of the "3.2.0 announcement":http://blog.chiliproject.org/releases/chiliproject-3-2-0-released/).
>
> I understand your desire of moving to a newer technology, it makes the development smoother, its interesting to keep up to date with new features, but think about your question: "Is the migration to a new technology worth it for the majority of users?"
>
> Be careful, it's very easy to end up moving from a cool technology to another and forget about users, which require new features, more ease of use etc... I understand it very well, I'm a developer too, marketing is there to makes me remember you "sells" features to users, not cool underling architectures :P
> It's a good think you keep up with technology but do not focus on that too much, this is just my humble suggestion.

There's some more parts to the switch than just "wanting new technology" :-) The most important reason to not stay on rails 2.3 is that support for it has been (or will shortly be, not sure what their exact schedule is without looking it up) discontinued, i.e. not moving would add taking (more) care of rails to our list of things to do. Ruby is another moving target here, rails 2.3 is compatible with 1.8.7 and 1.9.1, 1.8.7 will only receive security updates until next year and will be discontinued after that, so it's essentially something we have to move away from. That leaves 1.9.1, there's IIRC been some very subtle changes/bugs between 1.9.1 and the following 1.9.x versions, which we would have to watch out for ourselves if we stick to rails 2.3.

Other things beside that: it's becoming harder to find gems/plugins working with rails 2.3, there's things coming in future rails versions that we've been thinking about a lot (one example being the queue stuff being added to rails 4) that we could get "for free" and compatible with "the rest of the (rails) world" (i.e. not a ChiliProject-specific solutionâ€¦) and features/improvements in rails 3.1 and following should allow us to clean up more code (ARel should make the query model a lot cleaner for example), and so on. The last thing here, and that might far fetched, but I've had already 2 people telling me they'd be interested in getting involved but aren't "accustomed" to rails 2.3 anymore, thus they're keeping out for the moment.

2013-01-08 01:34 pm - Chris Dähn
I just wanted to remember that this ticket is quite important for some (mostly commercial) users.

We struggle with working around the problem of only one repo per project for nearly one year now... So: How are the chances to bring this feature into CP? For Trac this was a very often used feature - and solving that by introducing cross (sub)project rickets, roadmaps, wikis etc. is just a not very smart workaround.

Solving the missing feature by plugins could be quite hard or impossible?

How can we go further?