Quality Team - How to reduce the workload

We use https://gitlab.com/groups/tikiwiki/-/merge_requests instead of previous process. Information below is kept for posterity.

Context

Current (2011) QT process provides very high quality but is time consuming and there is a backlog. The good news is that there are a lot of commits. The bad news is that there are more commits than the QT can deal with.

This is a common challenge of FOSS projects. Here are related links from MediaWiki:

- http://www.mediawiki.org/wiki/Code_review_management/status
- http://thread.gmane.org/gmane.science.linguistics.wikipedia.technical/53884/
- http://thread.gmane.org/gmane.science.linguistics.wikipedia.technical/56748

How can we reduce the workload for the Quality team? (Ideally, **not** by reducing the number of fixes that are sent it)

On one hand, the number of commits should not grow any more because 6.x is getting old. On the other hand, 9.x will soon be with us and we need to have a sustainable system. Please note that Quality Team process requires time from QT members but also the community in general. So we should take this into account as well.

Topics

Reduce the scope

In the early days, there was branches/proposals/4.x and branches/proposals/3.x Then, it was reduced so the QT would review the stable branch (after x.1) but I think it's clear to everyone that this is still way too much work.

Officialize that Quality Team is only responsible for LTS

Accept	Undecided	Reject
6	0	0

- xavi
- jonnybradley
- pkdille
- changi67
- luci
- marclaporte

Branch management

Right now, the community is managing 4 branches LTS, LTS-proposals, stable and trunk.

Having a proposals branch causes more work: Translations and security commits need to be done in both since dogfooders are using proposals. Since no on is actively merging commits between the two branches, some translations can be lost.

The benefit is that security releases can be done quicker, even if not all proposals have been approved.

But this is just a band-aid to cover another issue. If there is a backlog, it's that the overall system is not sustainable. If we find a way to stay up to date, this shouldn't be too much of a challenge...

The benefit is that if a commit is approved (which is 90%+ of the cases), there is nothing more to do.

Eliminate the proposals branch and adapt the other guidelines to make this a good overall solution

Accept	Undecided	Reject
6	0	0
• xavi • jonnybradley • pkdille • changi67 • luci • marclaporte		

A mandatory delay (buffer time)

The doc says "*These proposed fixes should be done as a single commit (even if the original work in trunk took several goes to get right) so the whole change can be reviewed in one go, and if needs be, rolled back cleanly.*" However, we have seen many examples of this not being respected.

Also, sometimes, I get the impression that less experienced coders get more feedback and guidance when committing to the LTS branch. This can lead to multiple commits to address an issue and it gets messy. It also forces Quality Team members into becoming mentors. This is really nice of them but if this extra work is causing a backlog on the other commits to be a reviewed, this is not a good trade-off. Also, we could gently warn contributors whose commits are too often rejected.

Nothing stops the user to keep this commit for a week on his/her LTS instance before committing.

Add a general guideline that a commit should live in trunk/stable for at least a week before being committed to LTS

Accept	Undecided	Reject
0	3	2
	• jonnybradley • pkdille • marclaporte	• xavi • luci

We should use common sense though. Sometime, a really small fix with 0 risk of regression may not need that delay.

Approval threshold (number of votes)

Most of the time, negative votes are more "discussions on how to make it work" vs actual rejections (so delay above will be helpful as well). Do we really need 3-6 of our best developers to review all the commits? Historically, QT members have been voting things like "looks OK". The reason is that they feel compelled to vote on all commits. QT members are awesome but no one can know all the code as Tiki is the FOSS Web Application with the most built-in features. It's better for people to mostly review parts of the code they are comfortable with.

Quality has a price. And if we only have X number of votes, they should be more evenly spread out to all

the commits (instead of a backlog)

For one positive vote to be sufficient, as long as there are no negative votes.

Accept	Undecided	Reject
2	3	2
• luci • marclaporte	• xavi • jonnybradley • pkdille	• sept • changi67

How about two positive votes to be sufficient, as long as there are no negative votes?

Accept	Undecided	Reject
2	0	0
• jonnybradley • marclaporte		

Tracking the votes with Dogfood

a) The current process (voting on the mailing list) leads to some commits being checked several times, and others, none.
b) The process to track things is not in Tiki, and not publicly accessible so we are not dogfooding and we are not in a position to get outside help.

If this number goes above X, an automated email alert is sent to tiki-devel so more people can pitch in **before** it gets out of hand.

Would be nice to have chart of evolution (accepted vs rejected)

See Code Review

Setup code.tiki.org with a voting system and a dashboard so everyone in the community knows how many commits haven't yet been approved

Accept	Undecided	Reject
6	1	0
• xavi • jonnybradley • changi67 • dcedilotte • luci • marclaporte	• pkdille	

Having a timeframe

Without a reasonable timeframe to approve commits, we discourage people to follow the system.

As we have seen with our release process, if there is not set schedule, work piles up and it is just more

work to do it later. So we setup a 6-month release schedule and have been respecting it for over 3 years.

Standard time frame

Set x weeks as a standard time frame to review commits

Accept	Undecided	Reject
7	1	0
• jonnybradley • pkdille • sept • changi67 • xavi • dcedilotte • marclaporte	• luci	

Time frame status on a dashboard

Proposal 2: Setup code.tiki.org to indicate the number of commits that are beyond the time frame.

This will also help us maintain the "release at any time" status of the LTS branch.

Setup the code.tiki.org to indicate the number of commits that are beyond the time frame.

Accept	Undecided	Reject
8	0	0
• jonnybradley • pkdille • sept • changi67 • xavi • dcedilotte • luci • marclaporte		

Statute of limitations clause

Proposal 3: If we are beyond the time frame, and the commit doesn't have any negative votes (ex,: +1 and no minuses), shall we assume it's OK?

Commits that have not been treated by the quality team (according to the voting policy threshold) within the time frame are assumed to be OK and accepted.

Accept	Undecided	Reject
1	0	2
• marclaporte		• luci • jonnybradley

Open to more contributors

By having a dashboard, it makes it easier for anyone to contribute. Although the QT remains responsible for the overall process, having more voters reduces the workload

Let all members of the community vote, although their vote is non-binding

Accept	Undecided	Reject
4	2	1
• xavi • jonnybradley • luci • marclaporte	• pkdille • changi67	• sept

Managing reverts

Reverting is a pain. And if there is too much delay, the commit can get muddled with another and it's even more work.

Committing to LTS is a privilege.

Reverting commit is done by the committer (and in stable branch/trunk as well) . Uncooperative committers are warned and be removed that privilege

Accept	Undecided	Reject
0	3	4
	• xavi • jonnybradley • marclaporte	• pkdille • sept • changi67 • luci

So QT work is reduced to rejecting the commit and making sure it is indeed reverted.

Guidelines on appropriate contributions

In some projects, LTS is just for security fixes, and bug fixes are not allowed. In our case, there even has been new features added. (OK, it's not the first time we are unconventional!)

For feature additions: committers asks QT for permission before committing, especially if there is a DB schema change

Accept	Undecided	Reject
6	1	0
• jonnybradley • pkdille • sept • changi67 • luci • marclaporte	• xavi	

They could do this by indicating the revision number from the stable branch.

See Where to commit.

Have a review process when regressions slip through

The Quality has improved tremendously. But even with all that energy, a few regressions slipped in and caused some issues.

Have a quick review and basic tracking when regressions slip by

Accept	Undecided	Reject
2	3	0
• jonnybradley • luci	• pkdille • xavi • marclaporte	

So we'll know if our regressions go from the estimated current 2 or 3 per year to an unacceptable number

Dogfood LTS on as many sites as possible

Reducing the number of branches (above) will help with this as we'll converge our efforts.

Continue to run some *.tiki.org sites on LTS branch

Accept	Undecided	Reject
6	1	0
• xavi • jonnybradley • pkdille • changi67 • luci • marclaporte	• sept	

People that want utter stability can just wait for .x releases instead of randomly doing **svn up**...

Making sure commit is also in stable and trunk

The doc says: "The commit message on branches/proposed should contain the revision number(s) and commit message(s) from trunk where this was first committed."

Continue this excellent policy

Accept	Undecided	Reject
6	0	0
• xavi • jonnybradley • pkdille • sept • luci • marclaporte		

Also, we could make it easier for people to report issues via the dashboard.

If the commit is rejected in LTS, it may also mean commit should be reverted from trunk.

Officialize release responsibilities for LTS

With all the proposals above, the Quality Team responsibilities would be dramatically reduced. However, this one is crucial and the current situation leads to delays.

For the Quality Team to be responsible for LTS releases, including security releases

Accept	Undecided	Reject
6	1	0
• jonnybradley • pkdille • sept • changi67 • luci • marclaporte	• xavi	

Add a security-only phase in the LTS

For some LTS means security and bug fixes. For others, it means security only. We could maintain LTS period as security and bug fixes, and add an additional period, called "security-only" for security-only fixes

See: Version lifecycle

Adopt the idea of a 'security fixes' only period at the end of the LTS

Accept	Undecided	Reject
4	0	0
• luci • jonnybradley • xavi • marclaporte		

We will need to define what "security-only" means. Ex.: Experimental features are never supported.

Extend the stable period before the LTS period

Historically, LTS period starts 6-7 months after the release is stable. So when Tiki 7.1 is release Tiki 6x becomes LTS.

There can still be quite a few bug fixes and UI enhancements between months 6 and 12. We are thinking of adding a security-only phase to the LTS. See: Version lifecycle

Start LTS period after a longer stable period (ex.: 12 months instead of 6 months)

Accept	Undecided	Reject
1	1	1

- marclaporte
- xavi
- luci

Reduce the duration of the LTS period

LTS has been "bug and security" fixes. Assuming security-only phase in LTS idea is adopted, we could reduce the number of months of "bug fix" LTS and add more months of "security-only" LTS (which is way less work).

If "security-only" LTS idea is adopted, this could open up some possibilities

- Like reducing the LTS period from 18 months to 12 months and increase "security-only" LTS to 24 months.
  - So for some periods, there would be only be security fixes to review

See: Version lifecycle

Reduce the duration of the LTS period (ex.: 12 months instead of 18 months)

Accept	Undecided	Reject
2	0	2
• jonnybradley • marclaporte		• luci • xavi

Security-only phase in LTS and the Quality Team

With all the proposals above, the Quality Team responsibilities will be dramatically reduced. Security fixes don't happen very often (compared to bug fixes), but they can sometimes be tricky and introduce regressions. Thus, it's a priority to count on the vast experience of the Quality Team.

The time frame could be shorter than for non-security fixes.

The Quality Team reviews security fixes with the same process as for other fixes

Accept	Undecided	Reject
3	1	0
• jonnybradley • marclaporte • luci	• xavi	

alias

- Quality Team - How to reduce workload