Setting up a KDE Testing and Quality Software Team

I Introduction

In the past there were a few attempts to set up a Quality Team in KDE but they were not successful. Below I propose a few steps to start putting in place some infrastructure in order to better test our software. The idea is to start with the next 4.9 release and then pursue this for the port to Qt5 and the next KDE SC release.
The proposed actions target reinforcing (which means remain developers about them mostly) already existing infrastructures (such as unit tests, code check, review process for new inclusions) and setting new actions such as a more structured beta testing and a few areas for functional testing.

II Proposed testing methods for KDE SC 4.9

Goal : reinforce existing test infrastructure and setup new infrastructures. Follow the Release Schedule. Work with developers who want to be part of it, with the Bug Squad and with packagers and distribution people. Gather a team of enthusiast testers.

Reinforce : Code check: style, naming, …

(easy to improve)(each sub KDE project should indicate clearly which style they wish to follow)

- done by developers during the design of the software
- follow http://wiki.qt-project.org/API_Design_Principles
- can be JJ to fix indentation, variables and classes names
- use existing tools such as Krazy

Reinforce : Unit tests

(easy to improve)

- done by developers during the design of the software
- ensure that any lib outside kdelibs has tests (reinforce developers awareness of making the tests)
- reinforce awareness for developers to run the tests on a variety of platforms

Reinforce: review process

(easy to improve, done mainly by developers but testers can join to

Every new component coming into KDE SC should be in the review stage and its authors should send a mail to kde-core-devel, i18n, doc team, release team and relevant mailing list the program will join as well as a summary of the new app/lib plus the git/svn repository for reviewing. A commitment to maintain the code should be explicit in the mail asking for inclusion.

- check coding style
- check i18n (done by Albert Astal Cid)
- check relevant doc (handbook, userbase and techbase pages)
- test app behavior

See http://lists.kde.org/?l=kde-core-devel&m=121752904404479 for ideas that were suggested to implement better review.

NEW: Put in place structured beta-testing

Timeline: starts at the first beta release until the final 4.9.0 release (May 24th 2012 to August 1st 2012)
http://techbase.kde.org/Schedules/KDE4/4.9_Release_Schedule
The beta testers install the software and use it as they wish, with the understanding that they will report any errors revealed during usage back to the development organization. The beta-tester installs the beta software and uses it as he wishes. Collaboration with distributions which make beta packages available. Goal: that the software does what the user expects it to do.

The testers will report any bug and error revealed during usage using KDE bugzilla.

NEW: Put in place in a few selected areas functional testing

(black box testing also known as functional software testing, easy to implement)

A few areas (apps or whole "module") are defined with the involvement of the developers. Testers apply to test one piece of software (a plasmoid, an app, a kcm,..)
Check every UI element and see if it acts as you expect it to. Bug can be that the UI element does not do what it should (Quit does not quit for example) or you don't understand the purpose of the element, etc

III Methodology

A wiki page will list software and their beta-testers. Developers should be very responsive to fixing, assessing the errors found during the beta tests and should register their software prior to the test on the wiki page. The beta testers will apply there and agree to send prompt feedback about their test. Beta testers can be new users of the product, users of similar products from the competition or experienced users of prior versions. Ideally all platforms should be tested on. They should get a quick course on how to report bugs (steps to reproduce the bug in particular). This should be available before calling for beta-testers.
Beta-tests are conducted over a short period of time (during the beta releases).
Pro: identification of unexpected errors
Cons: not a thorough test as beta-testers only test what they use. Errors are sometimes reported without much details.
Focus will be on:

- incorrect or missing functionality
- interface errors
- errors in data structure used by interfaces
- behavior or performance errors
- initialization and termination errors

People doing those beta-tests will have the official status of "KDE 4.9 Beta-testers" which identify them as active contributors within the community. They are a bridge between the future users and developers.
Incentive for beta-testers sending the most valuable feedback?
Distributions will play a major role in these beta tests: define how to work with them and ensure having people from every distribution willing to help.

Specialized areas that some testers can focus on: i18n (bad layouts, wrong strings, missing context,...), RTL languages widgets behavior, usability, HIG rules

Follow-up to this work is to intensify bug triaging and confirm reports from beta-testers by the KDE bug squad.

Set up a few week-ends to test and triage and tackle bugs through IRC

IV TODO

- setup a dedicated mailing list: kde-testing (DONE: https://mail.kde.org/mailman/listinfo/kde-testing)
- set up a wiki page with a quick “HOWTO report bugs efficiently” (especially steps to reproduce).
- set up a wiki page with links to distributions repositories for KDE 4.9 Betas – add a HowTo build from tarballs link
- set up a wiki page with the goals of Beta testing
- set up a wiki page with programs whom developers are willing to get involved for Beta-Testers to register
- set some dates for IRC meetings
- write a Dot story explaining the process
- alert all social networks and relevant websites (blogs on PlanetKDE) to spread the news and get lots of testers

V Proposed goals for the next releases

- more automated tests (KDE Telepathy and Nepomuk have some)
- GUI automated testing (investigate software)
- extend areas to test with functional tests