Bug Tracking and Software Quality Assurance in a Free Software Project – the XEmacs Perspective

Gunnar Evermann

June 24, 1999

1 Introduction

In this paper I will discuss some aspects of Software Quality Assurance and how the current practice in the XEmacs project can be improved. In all complex (commercial) software project *software engineering* methods are routinely employed to ensure that the finished product fulfils the original requirements. In Free Software Projects this still is not as common as in commercial development efforts.

I will also discuss some aspects of the recently established Bug Tracking System that will help us in the future to ensure that a greater proportion of the problems encountered by the end users get fixed rapidly.

2 What is QA?

Quality Assurance is generally understood to comprise all activities that ensure that a contractually acceptable product is delivered. The concept of QA is applicable not only to Software but to any kind of development or production process. An international standard has been established to asses the quality process employed during the development of the product (ISO 9000).

Important criteria which should be used to evaluate the quality of a (software) product include:

- usability
- efficiency
- reliability
- maintainability
- testability

The relative weighting of these often conflicting criteria depends on the concrete application. In the (often mentioned) example of a control system for a nuclear reactor the reliability is obviously of paramount importance while the other criteria are of lesser importance. For software systems maintainability has proven to be extremely important as systems tend to stay around for much longer than originally planned (the best example of this is the current Y2K frenzy, where thousands of programmers have to work through millions of lines of undocumented Cobol source code written generations ago).

Organisations can have their quality process certified by the respective national standards body. This is generally a rather complicated (and therefore expensive) affair as it involves documenting every aspect of the whole development process.

3 Is Free Software different?

Many of the software engineering techniques developed for larger commercial software development can also be employed in a Free Software project but there are certain problems due to the different nature of the development process. A very good overview of the software engineering techniques applicable to a Free Software project is given by Paul Vixie in [1].

In a Free Software project most of the development work is done by volunteers who invest their spare time. Typically these are graduate students at universities around the world. These developers usually work on those aspects of the project that they consider *fun*. A consequence of this is that generally the documentation is neglected as the actual *hacking* is much more exciting. This problem is made even worse by the fact that the developers can drop out of the project at any time (e.g. if the amount of spare time available to them changes radically due to job related or personal reasons). The result is quite often pieces of (sometimes brilliant) code which may work very good at the moment but are extremely hard to maintain.

A big advantage of many volunteer based projects is that the developers are users of their system at the same time. Many problems can thus be discovered early on in the development process.

A problem is that the end user community cannot really exert any pressure on the developers as they don't have any contracts or such. The developers therefore can (and sometimes do) ignore the requests of their user base to fix bugs or extend the system in certain directions. In practice this is not quite so bad as many volunteers at some point develop some kind of *pride* in their product and therefore fix problems quite fast. Also the availability of source code enables even pure end users to investigate and resolve problems and thus become developers themselves.

4 XEmacs development

The development process in the XEmacs project is very open in the sense that new programmers or beta testers are always welcome and can easily get access to the current source code. Most of the discussion about the development takes place on the mailing-list `xemacs-beta`. In this forum many bugs or design problems are discussed and fixed early on. The quality of the bug reports is generally very high as the beta testers are in general technically very competent. Quite often bug reports are immediately accompanied by a suggested fix.

Beta releases normally occur very frequently and thus beta-testers who are not programmers themselves can help the developers by spotting problems early after changes are made. Public releases occur about once a year.

5 Bug Tracking

Every bug that remains in the publicly released version has suddenly a much greater impact than before as it now affects thousands of users many of which are not able to diagnose or fix such bugs on their own. When XEmacs crashes it writes a message to the console politely asking the user to report this crash to the developers. To handle these reports we recently installed a bug tracking system (see: `http://cvs.xemacs.org/XEmacs-bugs/`) based on the jitterbug system developed by the Samba team. All reports are evaluated, categorised and if possible fixed. Since the installation of this system in August 1998 more than 700 bug reports have been received.

Alas the quality of these reports is quite often rather appalling. Common problems are:

- Most (l)users don't actually read the nice message we print on their console when the crash occurs and do *not* follow the instructions on how to obtain a backtrace, instead they just copy this message verbatim.
- At many sites the MUAs or MTAs are completely broken resulting in mails which do not contain any valid addresses to reply to. This is extremely annoying and costs us a lot of time.
- Very old versions are still routinely used (e.g. in my research group there still exists an installation of Lucid Emacs 19.10 which was released five years ago!)
- XEmacs runs on many different Operating Systems and in many different configurations. Quite often no developer has access to the necessary platform.
- The best we can normally hope for is a C stacktrace of the crashed process, but many people have either disabled core dumps or are apparently unable to follow simple instructions.

On the positive side it must be said that some users are very helpful and provide very detailed descriptions on how to reproduce their problem. In these cases it is usually possible to fix the bug.

6 Testing & Documentation

An area that definitely still needs some further work is the documentation of the XEmacs implementation. The *Internals Manual* is pretty good but some modules are covered only very briefly.

Quite often the only form of documentation available is the actual source code itself, which was written by people who no longer contribute to the project. This is a somewhat frightening development as new programmers need to initially spend a lot of time to understand the existing (legacy) code before they can fix bugs or add extensions.

Experience with complex (commercial) software projects has shown that *testing* is a very important part of the development process. The appropriate attitude in my opinion is that every bug that is discovered in our released version also shows a failure in our testing efforts to cover the particular circumstances that trigger this bug. The testing obviously needs to be automated, so that it can be routinely executed on a wide variety of configurations. Also every fix of a bug should be accompanied by an appropriate addition to the test suite that exposes this bug in the old version and shows that it is fixed in the new version.

7 specific Mule problems

The MULE support in XEmacs suffers from all of the problems mentioned above. Also there is a lack of developers who are actually working on improving the integration. This seems to be related to the small number of developer who actually use a MULE enabled version of XEmacs for their daily work. Many problems are thus not discovered and the general usability of the multi-lingual features suffers. This again seems to deter new users from using the MULE support. Instead they tend to turn to FSF Emacs.

8 Conclusions

In the last years there has been a significant improvement in the techniques used to control the development process in the XEmacs project. The use of CVS for Version Control, the Bug Tracking System and the recent addition of automated tests are very positive and promising developments. There still is a lack of documentation and testing for certain modules of XEmacs (notably the redisplay engine) but hopefully these areas will be improved in the future.

References

[1] Paul Vixie. Software engineering. In Chris DiBona, Sam Ockman, and Mark Stone, editors, *Open Sources*. O'Reilly & Associates, 1999.