Evaluating Process Quality in GNOME based on Change Request Data

Holger Schackmann, Horst Lichter
Research Group Software Construction, RWTH Aachen University
{schackmann|lichter}@swc.rwth-aachen.de

Abstract

The lifecycle of defects reports and enhancement requests collected in the Bugzilla database of the GNOME project provides valuable information on the evolution of the change request process and for the assessment of process quality in the GNOME sub projects. We present a quality model for the analysis of quality characteristics that is based on evaluating metrics on the Bugzilla database, and illustrate it with a comparative evaluation for 25 of the largest products within GNOME.

1. Introduction

The GNOME project is structured into almost 400 products which include 38 products that have been deprecated[1]. GNOME imposes some guidelines for development, like the workflow for change requests, a common versioning scheme, a six month release cycle, and time lines for API freezes. However most of these guidelines are not considered to be mandatory, they can be adapted based on the needs of each product.

In order to analyze the quality of the change request process it is necessary to decide which requirements are explicitly or implicitly imposed on the change request process. Some explicit guidelines documented in the GNOME project are the Bug Writing Guidelines[2], the Bug Triage Guide[3], and the documentation of the status workflow in Bugzilla[4]. Thus the process involves the following roles and responsibilities:

General users report new change requests (**CR**), either a defect report or an enhancement request, in the status *Unconfirmed*.

Bugsquad members (i.e. the GNOME quality assurance team) triage CRs, either by confirming the CR (moving it to status *New)*, asking the reporter for more information (status *Needinfo*), or marking the bug either as *Duplicate*, *Invalid*, *NotGnome*, *NotABug* or *Obsolete.*

Developers/Maintainers are allowed to set the target milestone for a CR, to mark a CR as *Fixed* after committing changes to the source code repository, and to mark CRs with *WontFix*. They can also report new change requests directly in status *New.*

However, these rules have evolved over time, e.g. all products uniformly applied *Unconfirmed* as default state of new CRs not until May 2005. Since December 2005 developers were again allowed to report CRs directly in status *New*.

Quality characteristics can also be determined based on what is generally considered as good practice in open source projects [1], for example:

- The project should try to acknowledge each issue the moment it appears
- No conversations should take place in the bug tracker. Mailing lists are more appropriate.
- The database should be kept sane by finding obsolete CRs.
- The project should have a transparent development process, e.g. it must be visible which CRs are available with a milestone, to give users an incentive to move to this milestone [2].

In the next section we formulate the questions addressed in order to analyze the change request process. The evaluation approach and the applied tools are described in Section 3. Results are discussed in Section 4.

2. Questions addressed

Quality characteristics of the CR process can be defined along the roles mentioned in section 1. Each quality characteristic is described by one or several questions.

A. Quality of the CRs reported by general users:

- What is the quality of the CRs reported by general users in terms of completeness and redundancy freeness?

B. Quality of the CR triage by the Bugsquad:

- How long does it take for CRs to be triaged?
- Are triaged CRs correctly classified?

[1] http://bugzilla.gnome.org/describecomponents.cgi
[2] http://bugzilla.gnome.org/page.cgi?id=bug-writing.html
[3] http://live.gnome.org/Bugsquad/TriageGuide
[4] http://bugzilla.gnome.org/page.cgi?id=bug-status.html

C. Quality of Processing of CRs by the Developers/ Maintainers:

- How long does it take to resolve a CR?
- How often has a fixed CR to be reopened?
- How transparent is the availability of new features and bug fixes?

In the following we will analyse how these quality characteristics differ between GNOME products, and how they evolved over time.

3. Approach and tools used

Our approach is based on calculating metrics on the change request data that can be used as indicators for the quality characteristics of interest. To calculate metrics we applied the open source tool BugzillaMetrics [3]. It allows specifying metrics in a declarative language. Thus metrics can be described precisely on a higher abstraction level, which simplifies the process of developing and validating metrics [4].

Based on the questions formulated in section 2 we derived a number of corresponding metrics that are listed in table 1 with brief descriptions. Each metric is normalized such that its results are not biased by factors like size or age of the product. Furthermore each metric is specified in a way such that minimal values are considered to be optimal. The precise and complete specification of each metric is made available on *www.bugzillametrics.org*.

These metrics can then be evaluated for a number of selected products and a given time interval. The value distribution of the results for each metric in a time interval gives an impression on how good the different products perform in general and how large are the differences between the products.

Moreover we want to aggregate these raw results to be able to assess the quality in each of the three categories given in section 2. Since the GNOME development process imposes no absolute goals for the outcome of these metrics, we prefer to use the metrics results of the GNOME products itself in a selected time interval as comparison data. This pragmatic approach enables us to assess a quality characteristic of the development process of a certain product relative to other GNOME products. Moreover it can be analysed how quality characteristics evolved over time.

In order to specify the quality model, we used the QMetric quality model editor and evaluation tool [5]. The quality model defines how the individual metric results of a product are aggregated in order to assess a quality characteristic. The QMetric evaluation tool performs an automatic evaluation of the quality model based on results of a metric tool like BugzillaMetrics.

Table 1. Metrics used as quality indicators

Id	Metric	Description
A.1	Duplicated CRs	Number of CRs marked as *Duplicate* relative to the number of all resolved CRs in a time interval.
A.2	Invalid CRs	Number CRs marked as *Invalid*, *NotABug*, *NotGnome*, or *Incomplete* relative to the number of resolved CRs in a time interval.
A.3	Defect reports without version	Number of reported defects without a version number relative to the number of all reported defects in a time interval.
A.4	Transitions to *NeedInfo*	Number of transitions into the *NeedInfo* status relative to the number of CRs created in a time interval.
B.1	Time in *Unconfirmed*	Median residence time in days of newly created CRs in the status *Unconfirmed.*
B.2	False negative triaged CRs	Number of triaged CRs with resolution *Duplicate*, *Invalid*, *NotABug*, or *NotGnome* that have been reopened, relative to the triaged CRs in a time interval.
B.3	False positive triaged CRs	Number of CRs that have been confirmed and later resolved with *Duplicate*, *Invalid*, *NotABug*, or *NotGnome*, relative to the number of triaged CRs in a time interval.
C.1	Time until fixed	Median age in days of CRs that change into the status *Resolved/Fixed.*
C.2	Reopened Rate of fixed CRs	Number of fixed CRs that are reopened, relative to the number of fixed CRs in a time interval.
C.3	Fixed without milestone	Number of fixed CRs with no specified target milestone relative to the number of fixed CRs in a time interval.

The evaluation of the quality characteristics is based on classifying each individual metric value according to the quartiles of the metric results for comparison products. Then a linear equation is used to aggregate the results. In detail this can be defined as follows:

Let

C_m be a set of values for a given metric m measured for a number of products used as comparison data,

$Q_i(C_m)$ the i-th quartile of C_m, $i = 1..3$

The quartile classification q of a metric value v_m with respect to the corresponding comparison data C_m is defined as:

$$q(v_m, C_m) = \begin{cases} 1 & Q_3(C_m) < v_m \\ 2 & Q_2(C_m) < v_m \le Q_3(C_m) \\ 3 & Q_1(C_m) < v_m \le Q_2(C_m) \\ 4 & v_m \le Q_1(C_m) \end{cases}$$

A quality characteristic QC with underlying metrics $m_1, \ldots, m_n$ can then be evaluated as:

$$e(QC) = \frac{1}{n}\sum_{i=1}^{n} q(v_{m_i}, C_{m_i})$$

Hence the evaluation of a quality characteristic is normalized to a number between 1 and 4 with the following interpretation:

- 4 indicates that the considered product performs better than 75% of the products used as comparison data for each of the underlying metrics
- 1 indicates that the quality is poorer than in 75% of the compared products
- 2.5 can be interpreted as average quality.

Of course the quality model editor allows expressing more refined models, e.g. a weighting of the different metrics, or finer structuring of quality characteristics as tree or DAG. The presented quality model was deliberately kept simple in order to illustrate the approach.

4. Evaluation results

For the analysis we selected 25 of the largest products based on the total number of CRs in the status *Resolved* with resolution *Fixed*, since this is usually related to changes in the source code. Moreover we required that a product started in 2005 or earlier, and the product is not deprecated. This ensures that all considered products have continuous activity in Bugzilla. The 25 selected projects comprised 56% of all CRs with status *Resolved/Fixed* and 66% of all CRs in GNOME Bugzilla. Analysis was based on a snapshot of the GNOME Bugzilla database from September 19, 2008.

To reduce effects caused by different phases in the release cycle of a product we used years as time intervals for the evaluation. The value distribution for the year 2007 is given in table 2. We will first summarize some observations on this data.

Regarding the quality of the reported CRs, most products have a large number of redundant or invalid requests (A.1 and A.2). Users are rather disciplined in specifying a valid version number (A.3). Additional information from the reporter is typically requested for around 20% of the CRs (A.4). Detailed analysis can provide further insights, e.g. the search application *deskbar-applet* has the maximum values for the metrics A.1 and A.3 which implicates that its users unreflectingly report new CRs.

Triage by the Bugsquad works apparently good for most products. Most CRs are triaged in less than one day, with a low percentage of false positives and false negatives.

The values for C.1 indicate that most of the CRs that require development activity are fixed relatively fast. Also the reopened rate of fixed CRs is acceptable. Using

Table 2. Value distribution of the metric results in the year 2007 for 25 selected GNOME projects

Metric Id	Unit	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
A.1	%	8.00	21.49	41.04	57.61	90.16
A.2	%	4.29	15.91	31.27	38.78	54.62
A.3	%	0.61	2.39	8.60	32.09	97.18
A.4	%	1.63	10.66	20.18	28.79	64.36
B.1	days	0.15	0.46	0.78	2.16	30.70
B.2	%	0.00	0.00	0.02	0.20	1.02
B.3	%	0.07	0.28	0.60	1.17	3.07
C.1	days	2.28	5.32	12.94	48.71	118.96
C.2	%	0.34	1.41	2.31	2.98	7.55
C.3	%	45.28	87.52	93.02	98.88	100.00

the target milestone to create transparency is neglected by most of the developers. Only in the products *GIMP* and *Pan* more than 50% of the fixed CRs are marked with a target milestone.

The aggregated results for the quality characteristics in the years 2002 to 2008 are shown in Figure 1. For the sake of readability we focus on seven products that have been selected based on the number of CRs in status *Resolved/Fixed*. We have chosen the year 2007 as comparison data, as it is the last year with complete data. It is important to note that the comparison data is fixed to a single time period, otherwise changes over the years would be difficult to interpret.

The quality of the CRs reported for the considered products is relatively stable over the years (Figure 1 A). Products for non-specialized users like the file manager *Nautilus* and the mail application *Evolution* have a lower quality of the reported CRs. Not surprisingly the CRs for technology platforms like the widget toolkit *gtk+* and the streaming media framework *GStreamer* have higher quality.

The quality of CR triage has improved since 2005 for most of the products (Figure 1 B). This is probably caused by many efforts to attract volunteers for the Bugsquad, like making the triage rules explicit, coordinating meetings of volunteers (i.e. 'bug days'), and improving communication between Bugsquad members and developers [6].

The quality of processing the CRs by the developers has no general trend (Figure 1 C). The projects *nautilus* and *gnome-control-center* have improved in the last years. A detailed analysis for *gnome-control-center* shows that the product developers succeeded in decreasing the reopened rate of fixed CRs, and reducing the backlog of pending CRs, thus shortening the time until a CR is fixed. In contrast the result for *gtk+* has declined due to longer resolution times, and fewer target milestones being set.

5. Threats to Validity

In general the following threats to validity exist for the described approach:

Data quality: The Bugzilla database can reveal some inconsistencies e.g. due to maintenance like restructuring of products, or importing data from other Bugzilla instances. We found that CRs of some GNOME products had an incomplete activity history, but this did not affect the 25 selected products. Moreover the database snapshot did not contain security defect reports (<0.1% of all CRs).

Validity of the underlying metrics: It must be carefully validated that each metric is a proper numerical characterization of the qualities of interest, and that the measurements can be compared between different products. To ensure this, we applied a systematic stepwise validation approach [4].

Homogeneity of Bugzilla usage: The interpretation of different CR attributes can deviate between different products. We tried to base the metrics on fields with a commonly accepted interpretation. Results can also be distorted if issues are reported on other channels, like mailing lists. However according to Villa [6] using GNOME Bugzilla was widely accepted as standard in 2003.

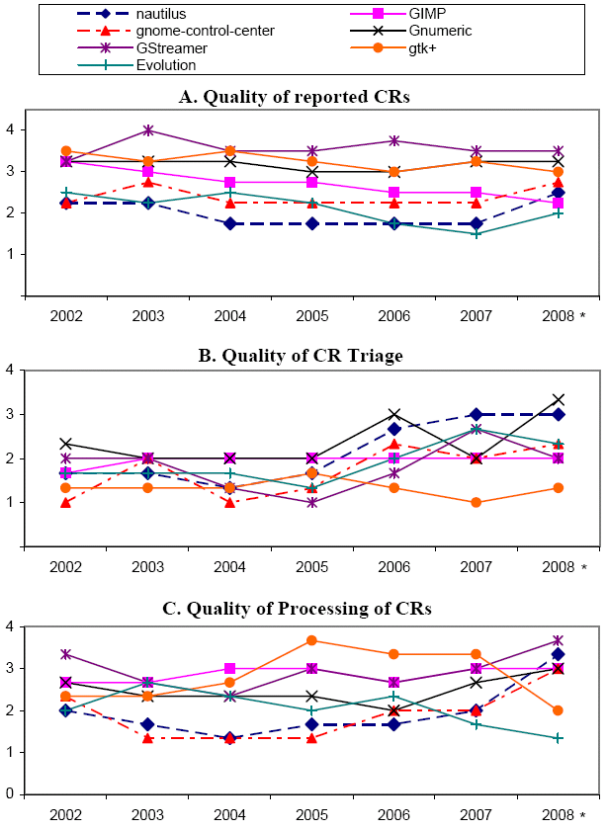

Figure 1. Aggregated results for the quality characteristics (*data for 2008 until September 19)

6. Conclusions

The usage of routinely collected change request data for the assessment of process quality is non-intrusive, and takes the past history of the process into account.

The presented approach is fully tool-supported [5]. Using declarative metric specifications facilitates to define the underlying metrics with precise semantics.

Aggregation of the results on the level of quality characteristics facilitates a condensed overview while preserving an intuitive interpretation due to usage of comparison data of real projects. Detailed analysis of the metric results can give valuable advice to the team members on realistic potential for improvement. It also allows to evaluate the effect of improvement activities. Further on such a quality model can be complementary to approaches for quality evaluation of open source software [7].

7. References

[1] Fogel, K., *Producing Open Source Software*, O'Reilly Media, Sebastopol, CA, 2005.

[2] Gamma, E., "Agile, Open Source, Distributed, and On-Time – Inside the Eclipse Development Process", Keynote Talk, ICSE'05, St. Louis, Missouri, 2005.

[3] Grammel, L., Schackmann, H., and Lichter, H., "BugzillaMetrics: an adaptable tool for evaluating metric specifications on change requests", in *Ninth international Workshop on Principles of Software Evolution* (Dubrovnik, Croatia, September 03 - 04, 2007). IWPSE '07. ACM, New York, NY, 2007, 35-38.

[4] Schackmann, H. and Lichter, H., "Comparison of Process Quality Characteristics Based on Change Request Data", in *Proceedings of the international Conferences on Software Process and Product Measurement* (Munich, Germany, November 18 - 19, 2008). R. R. Dumke et al., Eds. LNCS 5338. Springer-Verlag, Berlin, Heidelberg, 2009, 127-140.

[5] Schackmann, H., Jansen, M., Lischkowitz, C. and Lichter, H., "QMetric - A Metric Tool Suite for the Evaluation of Software Process Data", in *Companion Proceedings ICSE'09* (Vancouver, Canada, May 16-22, 2009), ACM, New York, NY, 2009.

[6] Villa, L., "Large free software projects and Bugzilla", in *Proceedings of the Linux Symposium* (Ottawa, Canada, July 23–26), 2003, 447-456.

[7] Samoladas, I., Gousios, G., Spinellis, D. and Stamelos, I., "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation", in *Open Source Development, Communities and Quality* (Milano, Italy, September 7-10), IFIP vol. 275, Springer, Boston, MA, 2008, 237-248.