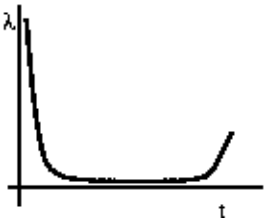
by Ralf Wieland
<rwieland/at/zalf.de>

About the author:

I develop a 3 dimensional simulation system of Eco and environment simulation (SAMT). It is based on Fuzzy logic, neural networks and cellular automates. It runs under Linux. I use Linux since version 0.99pl12.

Faulty Software



Abstract:

Controversial discussions have started around estimates about the number of faults which a given software has. Often the fault–density is used as a measure for the quality of software. Is that correct?
Using a simple model I will explain the different strategies and their consequences.

Translated to English by:
Guido Socher (homepage)

Faults in Open Source Software versus fault in Closed Source Software

The question whether Open Source Software or Closed Source Software is better is being heavily discussed in various computer magazines. The results are completely different dependent on which side the study ordered. The users of the software and the readers of the magazines don't really get anything out of those discussions. They just cause confusion and frustration. The interest in this actually potentially very interesting question quickly goes away.

The situation reminds me of the hardware and cpu benchmarking times. Every new "test" brought up new computers which would beat older models by far but when you actually bought such a computer the result was not the same.

Today the same discussions start in the software fault battle field.

I will try to shed some light into the situation using a simple and understandable model.

A simple model

Open Source Software and Closed Source Software contain faults. I assume that the programmers on both sides are equally good and produce therefore the same average fault density. Closed Source software development companies will employ testers to shake out the bugs. This is beneficial for them to some degree from a economic point of view and because customers may take them to court if essential functions are not working.
In the case of Open Source Software the pressure is not so high. The GPL excludes any warranties and responsibilities. Despite this the quality and security of Open Source Software tends to be very high.

For my model I assume that both closed and open source software implement the same functionality and both contain 100 faults.

For the first simulation I assume the following. The software company finds and cleans 20 faults using extensive tests. The open source developer hands out the software as early as possible and does not use extensive tests. To compensate the missing testing phase the Open Source developers work closely with the users and correct 80% of the faults. (100% is even withe Open Source not possible. Sometimes also new faults are added while correcting faults). Because of the availability of the source code about 10% of the faults are corrected per month.

The situation is quite different for closed source. It is more difficult to find the faults. For the model 8% of the faults are being reported (a very optimistic value). For the closed source company the process of correcting faults is usually expensive. The company will therefore try to correct only the important faults. It is of course a model and I am very interested to use one day real data.

	Open Source			Closed Source		
Month	errors	reported	corrected	errors	reported	corrected
1	100	10	8	80	6,4	3,2
2	92	9,2	7,4	76,8	6,1	3,1
3	84,6	8,5	6,8	73,7	5,9	2,9
4	77,9	7,8	6,2	70,8	5,7	2,8
5	71,6	7,2	5,7	67,9	5,4	2,7
6	65,9	6,6	5,3	65,2	5,2	2,6
7	60,6	6,1	4,9	62,6	5	2,5
8	55,8	5,6	4,5	60,1	4,8	2,4
9	51,3	5,1	4,1	57,7	4,6	2,3
10	47,2	4,7	3,8	55,4	4,4	2,2
11	43,4	4,3	3,5	53,2	4,3	2,1
12	40	4	3,2	51,1	4,1	2

If you look at the number of faults after the programs were in use for one year the the Open Source variant has a slight advantage. After half a year the fault rates are equal. In other words the users of Open Source software have to expect a lot more patches and corrections in the beginning. The amounts of updates will the cause higher cost. After this stabilization period the situation changes. Now the Open Source software has better quality and causes less costs due to updates.

In a second simulation I started with equal conditions for both Open Source Software and Closed Source software. Both did undergo extensive tests before the official release.

	Open Source			Closed Source		

Month	errors	reported	corrected	errors	reported	corrected
1	80	8	6,4	80	6,4	3,2
2	73,6	7,4	5,9	76,8	6,1	3,1
3	67,7	6,8	5,4	73,7	5,9	2,9
4	62,3	6,2	5	70,8	5,7	2,8
5	57,3	5,7	4,6	67,9	5,4	2,7
6	52,7	5,3	4,2	65,2	5,2	2,6
7	48,5	4,9	3,9	62,6	5	2,5
8	44,6	4,5	3,6	60,1	4,8	2,4
9	41,1	4,1	3,3	57,7	4,6	2,3
10	37,8	3,8	3	55,4	4,4	2,2
11	34,8	3,5	2,8	53,2	4,3	2,1
12	32	3,2	2,6	51,1	4,1	2

The situation looks now much better for Open Source. An extensive test is important. This is also the context in which you need to see this ZD−Net report: "A lack of commitment to testing by the Linux community may ultimately threaten the stability of the operating system, Linux kernel co−maintainer Andrew Morton has warned."
He sees that the test phase in no longer secured and to let the bulk of the test go to the final users is only the second best option.

How to rate this model

This model is based on very simple assumptions. The rate at which faults are discovered is e.g assumed to be constant. In reality this will not be the case. A new program will be tested by very interested users and here a large number of faults should be found. After a wider number of users come in the amount of faults found per time changes. It can both increase or decrease. This depends on the amount of remaining faults, the activity and knowledge of the users. In later phases the rate at which faults are found drops not only because the number of remaining faults is less but also because the interest of the users drops.
The willingness of the users to patch or update will be reduced over time.

Reliability

Technical equipment (e.g machines) have statistically different failure rates over their live time. At the beginning the failure rate is high and after a "burn in" phase a period with very low failure rate follows. Towards the end of the designed life time the failure rate goes again up.

At first glance it seems that software must be different since it is improved all the time. This is however not the case. One of the main reasons for this is that the development team changes (in open source and closed source) software after a while. The main developer may have left the project. You can then easily get a curve which is similar to the ones we see with hardware.

Number of errors

The absolute number of errors in a software project is never known. It is only possible to estimate (e.g 1 error per 1000 lines of code). This estimate depends however on the type of software and the experience of the programmer.

May studies use the number of found faults per time as a measure. Those studies claim then that Closed Source Software is of better quality. This is however wrong. Important is how many faults are left. It is important for our project (www.samt−lsa.org) to include the users into the development process. The user gets often patches and updates. In exchange ideas for further development and error reports are coming back. This an optimal solution for a scientific software.

Conclusion

A simple model shows that Open Source Software is not automatically better. Open Source Software has because of the openness the potential to become better. Important is that this stabilization process takes time and requires that the user is ready to update the software at the beginning frequently.
A test phase is important for the stabilization of the software no matter whether it is Closed or Open Source. An Open Source tester has the source code available and the the possibility to investigate the cause of a fault in details. An error report of such a tester is therefore much more valuable.

I have tried to present this controversial subject as objective as possible. It would be very nice if the above theoretical model could be undermined with real statistical tests.

2005−06−27, generated by lfparser_pdf version 2.51