Fehler in Software

von Ralf Wieland
<rwieland(at)zalf.de>

Über den Autor:

Ich entwickle ein räumliches Simulationssystem (SAMT) zur Umweltsimulation. Dazu verwende ich Fuzzy−Modelle, Neuronale Netze, zelluläre Automaten. Alles erfolgt unter Linux, das ich seit Version 0.99pl12 nutze.

Zusammenfassung:

Die Frage nach der in einer Software enthaltenen Menge an Fehlern ist heikel und wird kontrovers diskutiert. Oft wird die Fehlerrate als Maß für die Güte einer Software genommen. Ist das korrekt? Anhand einer einfachen Modellrechnung werden unterschiedliche Strategien und Konsequenzen diskutiert. Der Artikel soll zu einer kritischen Sicht und zu einer Diskussion anregen.

Fehler in Open Source versus Closed Source Software

Die Frage, ob Open Source Software oder Closed Source Software besser ist, wird in den Computerzeitschriften hitzig diskutiert. Je nachdem welche Seite die Studie in Auftrag gab, fallen die Ergebnisse aus. Die jeweilige Gegenseite weiß um die Schwächen und so kommen die Antworten und Kommentare recht schnell. Der Anwender hat von diesen Studien und Diskussionen kaum einen Nutzen. Er ist eher verwirrt und das Interesse, an dieser an sich interessanten und vor allem praktischen Frage läßt nach. Die Situation erinnert an die Zeit als Benchmarks aktuell waren. Jeder neue Rechner schlug den Konkurrenten im Benchmark. Nur leider sah die Praxis anders aus. Der Nutzer wollte keine Benchmarks ausführen, sondern seine Arbeit flüssig erledigen. Heute ist diese unsinnige Benchmark Protzerei einer durchaus praktikablen Beurteilung gewichen. Die Schlacht tobt um die Fehler in der Software.
Ohne fundamentale Ableitungen und Untersuchungen, soll mittels eines einfachen und hoffentlich plausiblen Modellansatzes das Problem beleuchtet werden. Vielleicht kommt es zu einer Diskussion und zu Verfahren, die dann eine ähnliche Entwicklung wie bei den Benchmarks bewirken. Es wäre schön, könnte ich hier eine Anregung geben und ein Mosaiksteinchen zu einer kritischeren Sicht beitragen.

Ein simples Modell

Open Source Software und auch Closed Source Software enthalten Fehler. Ich gehe davon aus, dass die Programmierer gleich gut sind, somit die durchschnittliche Fehlerzahl/(1000 Zeilen Code) gleich hoch ist. Closed Source Entwickler sehen sich Kunden gegenüber, die u.U. auch gerichtlich gegen sie vorgehen können. Aus eigenem Interesse sind sie verpflichtet einen Teil ihrer Entwicklungszeit in Tests zu investieren.

Auch ist es marktwirtschaftlich nicht günstig, wenn eine Software nur so von Fehlern strotzt. (Leider kommt Letzteres trotzdem recht oft vor.) Bei Open Source Entwicklern ist dieser Druck in der Regel nicht so groß. Siehe hierzu die GPL, die jegliche Garantie– und Schadenersatzansprüche ausschließt. Trotzdem zeichnet sich Open Source Software durch hohe Qualität und Sicherheit aus. Hier kommt zum Tragen, dass durch die Weitergabe der Quellen, die Möglichkeit Fehler zu entdecken und zu beseitigen ungemein größer ist als bei ihrem Gegenpart. Um die Sache etwas anschaulicher zu machen, soll ein einfaches Modell genutzt werden. Eine größere Closed Source Software soll mit einer die gleichen Funktionen realisierenden Open Source Variante verglichen werden. Der Vergleich ist rein fiktiv, soll nur die möglichen Konsequenzen verdeutlichen. Beide Varianten sollen per Definition 100 Fehler enthalten. Diese Zahl dient nur als Anhaltspunkt. Im ersten Modellfall wird folgendes angenommen: Die Softwarefirma bereinigt ihre Software in umfangreichen Tests um 20 Fehler. Die Open Source Entwickler geben die Software möglichst früh weiter und verzichten auf eine umfangreiche Testphase. Dafür arbeiten die Open Source Entwickler sehr eng mit den Anwendern zusammen und korrigieren 80% der gemeldeten Fehler. (100% ist selbst bei Open Source nicht möglich, manchmal kommen auch neue Fehler hinzu!). Durch die Offenheit des Quellcodes werden monatlich im Schnitt 10% der in der Software enthaltenen Fehler entdeckt.
Ganz anders ist das bei Closed Source. Hier werden Fehler schwieriger entdeckt. Im Modellansatz sollen 8% der noch enthaltenen Fehler gemeldet werden (ein sehr optimistischer Ansatz). Für die Closed Source Firma ist die Beseitigung der Fehler mit erheblichen Kosten verbunden, da hier in der Regel auch andere Vertriebswege als das Internet bedient werden müssen. Die Firma wird also bestrebt sein, nur unbedingt notwendige Änderungen vorzunehmen. Im Modell sind das 50% der gemeldeten Fehler. Wie gesagt, es handelt sich um ein fiktives Modell. Falls aber genaue Zahlen verfügbar sind, wäre es interessant diese zu nutzen.

	Open Source			Closed Source		
Monat	Fehler	gemeldete	korrigierte	Fehler	gemeldete	korrigierte
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

Sieht man sich die Fehleranzahl nach der Grenznutzungsdauer des Programms von einem Jahr an, so ist die Open Source Variante leicht in Vorteil. Wichtiger als die absolute Anzahl der Fehler ist aber die Fehlerzahl während der Nutzungsdauer. Nach etwa einem halben Jahr sind im Beispiel die Fehlerzahlen gleich. Mit anderen Worten, die Nutzer der Open Source Software müssen in der Anfangsphase mit erheblich mehr Fehlern rechnen. Auch müssen sie viel häufiger Patchen, was zusätzliche Kosten verursacht. Ist aber die Software über ihre ersten Versionen hinweg, kehrt sich die Qualität um. Nun ist die Open Source Software günstiger.

In einem zweiten Modellfall soll bei gleichen Ausgangswerten auch die Open Source Software einen umfangreichen Test unterzogen werden. Beide Varianten gehen nun mit gleichen Fehlerzahlen an den Start.

	Open Source			Closed Source		

Monat	Fehler	gemeldete	korrigierte	Fehler	gemeldete	korrigierte
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

Jetzt sieht die Lage viel besser für die Open Source Software aus. Ein umfangreicher Test ist wichtig! In diesem Zusammenhang ist auch folgende ZD−Net−Meldung einzuordnen: " Andrew Morton, Kernel−Maintainer und enger Mitstreiter von Linus Torvalds im Linuxkernel−Projekt warnt vor möglicherweise zunehmenden Instabilitäten bei der Weiterentwicklung von Linux." Er sieht die Testphase nicht mehr gewährleistet. Es ist also auch für Open Source entscheidend, wie die Tests organisiert sind und wie effektiv sie durchgeführt werden. Eine Delegation der Tests an die Anwender ist nur eine zweitbeste Lösung.

Kritische Einordnung des Modells

Das Modell geht von sehr einfachen Annahmen aus. So wird die Fehlererkennungsrate als konstant angenommen. In der Praxis ist das aber nicht so. Ein neues Programm wird von Interessierten getestet. Hier wird eine große Zahl von Fehlern entdeckt. Nach und nach wird das Programm dann auch für die Allgemeinheit interessanter und die Zahl der entdeckten Fehler verändert sich. Sie kann sowohl zunehmen als auch abnehmen während dieser Phase. Das hängt neben der Anzahl der verbliebenen Fehler stark von der Anzahl, der Aktivität, dem Kenntnisstand etc. der Nutzer ab. In der Reifephase nimmt die Zahl der Fehlermeldungen in der Regel ab. Das ist aber nicht nur auf die Reife des Programms zurückzuführen, sondern auch auf das abnehmende Interesse der Nutzer, die Abwanderung von Nutzern auf Konkurrenzprodukte etc. Die Fehlererkennungsrate schwankt also im Laufe der Nutzung.
Ebenso ist die Patchwilligkeit der Entwickler nicht zeitlich konstant. Hier soll ein wichtiger Aspekt einmal angesprochen werden, die Badewannenkurve in technischen Zuverlässigkeitsberechnungen.

Zuverlässigkeit

Eine technische Einrichtung, sei es nun eine Werkzeugmaschine, ein Auto oder ein Rechner durchläuft während seines Betriebs unterschiedliche Phasen. In der ersten Phase, den sogenannten "Kinderkrankheiten", gibt es häufig Ausfälle. Diese sind auf Bedienfehler, fehlerhafte Teile, fehlerhaften Zusammenbau, fehlenden "burn in" Prozess etc. zurückzuführen. Nach einer gewissen Zeit verringern sich die Fehler und das System läuft lange zuverlässig. Zum Schluß der Nutzung stellen sich Abnutzungserscheinungen ein. Das System wird in dieser Phase wieder fehleranfälliger. Trägt man die Zahl der entdeckten Fehler über die Zeit auf, so entsteht das typische Bild einer Badewanne.
Bei Software scheint das alles anders. Es werden ständig Verbesserungen vorgenommen. Die Software müßte

also immer besser werden. Wird sie aber nicht! Sehr oft wird Software, die einmal recht gut lief durch Verbesserungen immer unzuverlässiger. Wie kommt das? Neben einer Reihe von Gründen scheint meiner Meinung nach ein ganz wichtiger Grund zu sein, dass der ursprüngliche Entwickler der Software nicht mehr die Entwicklung trägt. Oft hat er das Projekt verlassen, oder andere Aufgaben übernommen. Die Entwickler, die mit der Pflege vertraut sind, kennen nicht mehr alle Einzelheiten des Projektes. Die Weiterentwicklung des Projektes erfolgt weniger inhaltlich durch Integration neuer Methoden, sondern mehr oberflächlich durch Integration in andere Umgebungen, wie beispielsweise graphische Nutzerinterfaces. Mit dieser "Softwarepflege" nimmt die Zuverlässigkeit oft ab. Die daraus resultierende Kurve kann einer Badewannenkurve durchaus ähneln.

Fehleranzahl

Die absolute Anzahl der Fehler in einem Softwareprojekt ist nie bekannt. Sie kann aber aus Erfahrungswerten (z.B 1Fehler/(1000 Programmzeilen)) abgeschätzt werden. Leider ist diese Abschätzung von der Art der Software, vom Programmierer, den Werkzeugen etc. abhängig, so dass es nur grobe Anhaltspunkte gibt. Betrachtet man, wie in Studien immer wieder zu beobachten, die Anzahl der entdeckten Fehler pro Zeiteinheit, kann das ein völlig falsches Bild setzen. Anscheinend hat die Closed Source Software eine bessere Güte, da die Zahl der entdeckten Fehler kleiner als die bei der Open Source ist. Das wird von Firmen gern hervorgehoben, ist aber nicht zulässig. Entscheidend ist nicht die Anzahl bis zu einem Zeitpunkt entdeckten Fehler, sondern die Anzahl der verbleibenden Fehler. Letztere kann sehr unterschiedlich ausfallen, wie die Modellrechnungen zeigten. Sehr wichtig ist außerdem, wie schnell auf entdeckte Fehler reagiert wird und natürlich wie gefährlich diese Fehler waren. Bei unserem Projekt ist es wichtig, die Anwender in die Entwicklung einzubeziehen. Der Anwender bekommt häufige Patches und liefert Fehlerberichte und Hinweise, Wünsche für die Weiterentwicklung. Für die Entwicklung wissenschaftlicher Software kann dieser Weg optimal sein. Andere Projekte, die z.B. Internetapplikationen entwickeln, müssen viel mehr in die Sicherheit investieren, was eine umfangreiche Testphase voraussetzt.

Schlußfolgerungen

Die einfache Modellrechnung zeigte, dass Open Source nicht per se zuverlässiger ist. Sie hat durch ihre Offenheit das Potential besser zu werden. Zu beachten ist, dass dieser Prozeß der Stabilisierung Zeit benötigt und vom Anwender das häufige Einspielen von Patches erfordert. In der Regel ist Letzteres für den Nutzer von Open Source wichtiger als für den Nutzer von Closed Source. Wobei das auch wieder vom Umfeld abhängt. So können Fehler in vernetzten Microsoft Systemen allein durch ihre höhere Verbreitung eine größere Auswirkung haben als Linuxrechner.
Ein umfangreicher Test sollte unabhängig von Art der Software durchgeführt werden. Hier kann Open Source die bessere Lösung sein, da durch die Quelloffenheit der Kreis der Tester einfach erweiterbar ist. Ein Tester, der den Quelltext versteht und dem Fehler auf den Grund geht, leistet viel mehr als ein Nutzer der einen Fehlerbericht abgibt.
Der Bewertung nach der Fehlerrate sollte man kritisch gegenüber stehen. Nur wenn die Randwerte (wie sind die Fehler einzuordnen, wie wird auf sie reagiert, wieviele Nutzer sind betroffen, wie wird auf Fehlermeldungen von Seiten der Entwickler reagiert, wieviele Nutzer melden Fehler, werden die gemeldeten Fehler öffentlich etc.) bekannt sind, sagt die Fehlerrate etwas aus. Ein Verschweigen der Randbedingungen kann ein völlig falsches Bild vermitteln.
Ich habe mich bemüht einige Gedanken zu diesem kontroversen Thema zusammenzutragen und möglichst objektiv zu bleiben. Um zu wirklich beweiskräftigen Aussagen zu kommen, müssen die Abschätzungen durch Messungen unterlegt werden. Es wäre schön, könnten möglichst viele an diesen Messungen und Tests beteiligen und diese Tests statistisch auswerten. Damit könnte die Schwäche der obigen Modellrechnung, die

auf reinen Annahmen beruht, überwunden werden.

Der LinuxFocus Redaktion schreiben © Ralf Wieland "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Autoren und Übersetzer: de −−> −− : Ralf Wieland <rwieland(at)zalf.de>

2005−06−27, generated by lfparser_pdf version 2.51