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

L´auteur:

Je développe un système de simulation tri−dimensionnelle d'Eco et de simulation d'environnement : (SAMT). Il est basé sur la logique floue, les réseaux neuronaux et les automates cellulaires. Il tourne sous Linux, que j'utilise depuis la version 0.99pl12.

Logiciel défectueux

Résumé:

Des discussions controversées ont commencé sur les estimations du nombre de défauts qu'avait un logiciel donné. Bien souvent, la densité en défauts est utilisée comme une mesure de la qualité d'un logiciel. Est−ce correct ?
En utilisant un modèle simple, je vais expliquer les différentes stratégies et leurs conséquences.

Traduit en Français par:
Jean−Etienne Poirrier
(homepage)

Les défauts dans les logiciels Source Ouvert contre les défauts dans les logiciels à Source Fermé

La question de savoir qui, des logiciels Source Ouvert ou des Source Fermé, sont les meilleurs, a été abondamment discuté dans les divers magazines informatique. Les résultats sont complètement différents en fonction du côté qui commandite l'étude. Les utilisateurs de logiciels et les lecteurs de magazines ne gagnent pas grand chose de ces discussions. Elles génèrent juste plus de confusion et de frustrations. L'intérêt de cette question, potentiellement très intéressante, disparaît rapidement.

La situation me rappelle les temps des tests du matériel et du cpu. Chaque nouveau « test » apporte de nouveaux ordinateurs qui vont battre de loin les anciens modèles mais, lorsque vous achetez réellement un tel ordinateur, le résultat n'est pas le même.

Aujourd'hui, les mêmes discussions ont débuté dans le champ de bataille des défauts de logiciels.

Je vais essayer d'éclaircir un peu la situation en utilisant un modèle simple et compréhensible.

Un modèle simple

Les logiciels Source Ouvert et Source Fermé contiennent des défauts. Je suppose que les programmeurs des deux côtés sont aussi bons et produisent, en conséquence, la même densité moyenne de défauts. Les sociétés de développement de logiciels à Source Fermé vont employer des testeurs pour éliminer les bogues. D'un point de vue économique, c'est tout bénéfices pour eux, jusqu'à un certain point mais également parce que les clients peuvent les poursuivre en justice si les fonctions essentielles ne fonctionnent pas.
Dans le cas des logiciels Source Ouvert, la pression n'est pas aussi grande. La GPL exclut toutes garanties et responsabilités. En dépit de cela, la qualité et la sécurité des logiciels Source Ouvert semble être très grande.

Pour mon modèle, je suppose que tant le logiciel à Source Fermé que le logiciel à Source Ouvert implémentent la même fonctionnalité et que tous les deux contiennent 100 défauts.

Pour la première simulation, je fais les suppositions suivantes. La compagnie de logiciel trouve et corrige 20 défauts en utilisant des tests étendus. Le développeur Source Ouvert distribue aussitôt que possible et n'effectue pas de tests étendus. Pour compenser la phase de test manquante, le développeur Source Ouvert travaille en étroite collaboration avec les utilisateurs et corrige 80% des défauts (même avec le Source Ouvert, 100% n'est pas possible ; de temps en temps, de nouveaux défauts sont ajoutés lors de la correction de défauts existants). Grâce à la disponibilité du code source, environ 10% des défauts sont corrigés par mois.

La situation est très différente pour la partie à Source Fermé. Il est plus difficile de trouver les défauts. Dans ce modèle, 8% des fautes sont rapportées (une valeur très optimiste). Pour la compagnie à Source Fermé, le processus de correction des défauts est habituellement très coûteux. La compagnie va, dès lors, essayer de ne corriger que les défauts importants. Ceci est, bien sûr un modèle et je serai très intéressé d'utiliser des données réelles, un jour.

	Source Ouvert			Source Fermé		
Mois	erreurs	rapportées	corrigées	erreurs	rapportées	corrigées
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

Si vous regardez le nombre de défauts après que les programmes aient été utilisés pendant un an, la variante Source Ouvert a un léger avantage. Après une demi−année, le taux de défauts est équivalent. En d'autres mots, les utilisateurs de logiciels Source Ouvert doivent s'attendre à beaucoup plus de patches et corrections au début. La quantité de mises à jour va engendrer un coût plus élevé. Après cette période de stabilisation, la situation change. Maintenant, le logiciel Source Ouvert est de meilleure qualité et engendre moins de coûts dûs aux mises à jour.

Dans une seconde simulation, je suis parti de conditions égales, tant pour le logiciel Source Ouvert que le logiciel à Source Fermé. Tous les deux ont subi des tests étendus avant leur diffusion officielle.

	Source Ouvert			Source Fermé		
Mois	erreurs	rapportées	corrigées	erreurs	rapportées	corrigées
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

Maintenant, la situation semble beaucoup mieux pour le Source Ouvert. Un test étendu est important. C'est également dans ce contexte que vous devez voir ce rapport de ZD−Net : « Le co−mainteneur du noyau Linux, Andrew Morton, a averti qu'un défaut d'engagement dans les tests de la part de la communauté Linux pourrait, en définitive, menacer la stabilité du système d'exploitation ».
Il trouve que la phase de test n'est plus sécurisée et que laisser le plus gros des tests être réalisé par les utilisateurs finaux n'est que la seconde meilleure solution.

Comment quantifier ce modèle

Ce modèle est basé sur des suppositions très simples. Le taux auquel les défauts sont découverts est, par exemple, supposé constant. En réalité, cela ne sera pas le cas. Un nouveau programme sera testé par des utilisateurs très intéressés et, ici, un nombre important de défauts devrait être trouvé. Après qu'un nombre plus important d'utilisateurs soit arrivé à utiliser le logiciel, la quantité de défaut trouvé en fonction du temps change. Il peut aussi bien augmenter que diminuer. Cela dépend de la quantité de fautes restantes, de l'activité et de la connaissance des utilisateurs. Dans des phases plus avancées, le taux avec lequel les fautes sont trouvées chute, non seulement parce que le nombre de défauts restant est moindre mais aussi parce que l'intérêt des utilisateurs chute également.
La volonté des utilisateurs d'appliquer des patch ou de mettre à jour sera réduite au cours du temps.

Fiabilité

L'équipement technique (par exemple, les machines) ont, statistiquement, des taux d'échec différents au cours de leur temps de vie. Au début, le taux d'échec est élevé et, après une phase assez chaude, suit une période avec un taux d'échec très bas. A la fin de la durée de vie prévue, le taux d'échec augmente de nouveau.

A première vue, il semble que cela doit être différent pour les logiciels puisqu'il est amélioré tout le temps. Ce n'est cependant pas le cas. Une des raisons principales de cela est que l'équipe de développement change le logiciel après un certain temps (tant en Source Ouvert qu'en Source Fermé). Le développeur principal peut avoir quitté le projet. Vous pouvez facilement avoir une courbe similaire à celles que nous avons avec le matériel.

Nombre d'erreurs

Le nombre absolu d'erreurs dans un projet logiciel n'est jamais connu. Il est seulement possible de l'estimer (par exemple : 1 erreur pour 1000 lignes de code). Cette estimation dépend cependant du type de logiciel et de l'expérience du programmeur.
De nombreuses études utilisent le nombre de fautes trouvées en fonction du temps comme mesure. Ces études prétendent que les logiciels à Source Fermé sont de meilleure qualité. C'est cependant faux. L'important est le nombre de fautes qui sont laissées. Il est important pour notre projet (www.samt−lsa.org) d'inclure les utilisateurs au processus de développement. L'utilisateur reçoit souvent des patches et des mises à jour. En échange, des idées pour des développement futurs ainsi que des rapports d'erreurs reviennent. C'est une solution optimale pour un logiciel scientifique.

Conclusion

Un modèle simple montre que les logiciels Source Ouvert ne sont pas automatiquement meilleurs. Les logiciels Source Ouvert ont le potentiel de devenir meilleur, grâce à leur ouverture. L'important est que ce processus de stabilisation prenne du temps et requiert de l'utilisateur qu'il soit prêt à mettre à jour le logiciel fréquemment, dès le début.
Une phase de test est importante pour la stabilisation du logiciel, peu importe qu'il soit Source Ouvert ou à Source Fermé.
Un testeur Source Ouvert a à sa disposition le code source et la possibilité d'étudier la cause du défaut en détail. Un rapport d'erreur d'un tel testeur en devient ainsi beaucoup plus valable.

J'ai essayé de présenter ce sujet controversé de manière la plus objective possible. Il serait très intéressant si le modèle théorique ci−dessus pouvait être confronté à de réels tests statistiques.

Site Web maintenu par l´équipe d´édition LinuxFocus	Translation information:
© Ralf Wieland	de −−> −− : Ralf Wieland <rwieland/at/zalf.de>
"some rights reserved" see linuxfocus.org/license/	de −−> en: Guido Socher (homepage)
http://www.LinuxFocus.org	en −−> fr: Jean−Etienne Poirrier (homepage)

2005−07−08, generated by lfparser_pdf version 2.51