© *1997−2009, Millennium Mathematics Project, University of Cambridge.*
Permission is granted to print and copy this page on paper for non−commercial use. For other uses, including electronic redistribution, please contact us.

15/07/2003
News

Open wide...

BOOKMARK

The open source community may now have more than ideology on their side, with researchers showing mathematically that their "release early, release often" software development model is the quickest way to bug−free code.

The bazaar versus...
Image DHD Photo Gallery

Software is usually created in one of two ways. Eric Raymond coined names for the two approaches: the *cathedral* approach, where a team of programmers work as an isolated group creating the software, only releasing it to their users every so often after substantial changes; and the *bazaar* approach, where new versions of the software are released as often as possible, perhaps as a result of work done by a community of

user−programmers. Most proprietary software, such as Microsoft Windows, falls into the cathedral category, while the Linux operating system and Apache, the most popular web server, are examples of the bazaar approach.

In a paper currently being refereed, Damien Challet and Yann Le Du from the University of Oxford characterised *open source* software as that developed in the bazaar style, and *closed source* as that created with the cathedral approach. They developed a mathematical model of how bugs, or errors in the software, are detected and resolved for the two types of software development.

...the cathedral
Image DHD Photo Gallery

The model assumes that each software program consists of a number of independent parts, and that users will spot bugs present in a part of the program with a certain probability. Once a bug is reported, the chance that some programmer will be able to fix it is given by another probability. The model of open source software assumes that at each step, the users are using a new version of the software containing all the changes made by the programmers in the previous step. In contrast, for the closed source model, new versions of the software is released to users less frequently.

The study shows that under the same conditions − the same number of users with the same chance of spotting bugs, the same number of equally able programmers, and the same initial level of bugs − open source software will always become bug−free quicker than closed source software. For closed source software to become completely reliable in the same amount of time, the programmers must be more able, or there must be more of them, than for open source software.

The researchers explain that this is a result of the release cycles for closed source software. When a new version is released, after an initial rush of bug finds by users, the programmers are left to find and correct errors with little feedback until the next release. In contrast, the frequent releases of open source software allow for continuous testing by users, and feedback to programmers.

The researchers were also able to use their model to investigate the dynamics of real life projects, such as the development of the open source project Linux, by using records of the number of users and programmers, and the size of the program measured in lines of code. Since its birth in 1991, the growth of the Linux code has been approximately quadratic − much faster than the linear growth of other software projects such as Mozilla − and yet it has a reputation for being very reliable. According to the researchers, "this leaves the question of how Linux could grow at such a pace without compromising its quality".

The answer appears to lie in the quality of Linux programmers. The researchers had to put a lower bound on their ability − the probability that they would successfully fix a bug − to make the model emulate the development of Linux. The researchers concluded that "rapid software growth can indeed lead to high quality software, even in adverse conditions, provided that the programmers' quality is high enough."

The researchers are aware that there are limitations to their current model, including the assumption that a program is made up of independent parts. The next step is to apply the model to a *scale free network*, which Ricard Solé from the Complex Systems Lab, Barcelona, and his colleagues from the Santa Fe Institute, New Mexico, have shown to be a more accurate description of the structure of software.

A scale free network is a network of points, called *nodes*, joined together in such a way that most of the nodes are linked directly to only a few others, while an important few nodes are linked to many others. These highly linked nodes act like hubs, and a consequence is that the average distance between any two nodes in the network (the number of links you have to follow to get from one to the other) is small. Many real−world networks have recently been shown to be scale free, including the World Wide Web, with sites like yahoo acting as hubs, and the internet, the physical network of routers and wires behind the WWW − even some metabolic networks and food webs appear to be scale free.

So as research continues, many in the programming community will be waiting to see if it provides more ammunition for the open source debate. Meanwhile, software users will be hoping that it will lead to more reliable software, as we head towards an ever more computer−dependent future.

Rachel Thomas

Plus is part of the family of activities in the Millennium Mathematics Project, which also includes the NRICH and MOTIVATE sites.