30
Nov
2009

Seven Observations On Software Maintenance And FOSS

The November 2009 issue of Communications of the ACM (CACM) has a very interesting article by Paul Stachour and David Collier-Brown entitled "You Don't Know Jack About Software Maintenance". The authors argue energetically for using versioned data structures and "continuous upgrading" to improve the state of the art of software maintenance.

The piece got me thinking about FOSS (Free and Open Source Software) and "continuous upgrading". Here are seven observations on FOSS software maintenance that occurred to me as I reflected on the CACM article:

1. FOSS projects "continuously" apply bug fixes and feature enhancements at no additional cost to their users. By applying these improvements "continuously", the user reaps a steady stream of "interest payments" providing ever-improving security, performance, and functionality.
2. Since FOSS incurs no licensing or license management costs, upgrading FOSS is not hindered by capital expenses.
3. Typically support in FOSS projects is focused on the current stable version. Therefore, upgrading to the current stable version is the preferred way to receive the best support from FOSS communities.
4. One of the key reasons behind Debian's strong track record of "continuous upgrading" is its way of handling the tricky issues involved with dependent library upgrades (such as libc6, libssl.so.0.9.8, & etc). The chapter on Shared Libraries in the Debian Policy Manual details a proven method to effectively handle library upgrade issues (including its sophisticated handling of versions).
5. When upgrading is applied routinely and "continuously", it becomes crucial to support customizations across upgrades which can be one of the biggest obstacles to a smooth upgrade (see my earlier post on customization and upgradeability). One reason for Debian's effectiveness in this regard is its robust configuration file handling policy.
6. It is worth noting that the "continuous" implied here is not the one emphasized in dictionaries (which takes its nuances from the mathematical / physics concept of "no interruptions" and the epsilon-delta definition that students of Calculus learn). That concept of "continuous" is impossible in systems administration which is necessarily discrete as are all computer operations. The connotation required here is, perhaps, "unending", or "eternal" or somesuch.
7. The "right" frequency for "continuous" upgrades is a complex tradeoff between business requirements and upgrade infrastructure maturity. Debian and Ubuntu provide vary mature support for "continuous upgrading". They support the upgrade of production servers through release after release after major release with minimal downtime or risk of a glitch that could affect users. Their current release frequency of about 2 years may be the best we can do given the current state of the art of software maintenance. I hope we can learn to increase the frequency as better engineered upgrade policies are developed.

I prefer the name **“eternally regenerative software administration”** over “continuous upgrading”. It avoids the philosophical problems with the word “continuous” and emphasizes the active, “ecological” approach needed to envision the engineering of “regenerativity” in software. By that I mean software maintenance should involve building the system so each new version enables installation of the next while facilitating management of any customizations and integration with other software (including libraries and other “helper” applications). Regenerativity is the process of growth and change used by Nature itself. Software maintenance needs to follow similar principles.

Posted by CJ Fearnley *in* Debian, Eternally Regenerative Software Administration, Ubuntu, *0 comments*