16
Oct
2009

Customization, Upgradeability and Eternally Regenerative Software Administration

Mary Hayes Weier wrote an interesting article in this week’s edition of InformationWeek on "Alternative IT: CIOs are more receptive than ever to new software models". What is great about her article is how she captured the divergent views on IT models (such as SaaS, cloud computing, etc.) and gave nice vignettes of different organizations trying different parts of various models. I especially valued her use of cognitive dissonance to leave the reader thinking … better informed but without a firm conclusion.

There are so many parts of the article that I could blog about, but the one that touched the core of my thinking about *“eternally regenerative software administration”* was the quote by Bill Louv, CIO at GlaxoSmithKline, who said

> "And here’s the rub: When you customize software, it’s difficult to implement future upgrades from the vendor"

Louv touched the very bane of *eternally regenerative software administration*! Software should accommodate both customization and upgradeability: these two elements of software administration are at the heart of my notion of **eternally regenerative software administration**: how to preserve customizations and provide smooth (near zero downtime with almost no glitches) upgrades through major release after major release. It is a big challenge, but in our experience the Free and Open Source Software (FOSS) communities are at the leading edge in finding solutions to these conflicting objectives. Here are some of the innovative ideas from the FOSS world which should serve as models or design patterns for all software developers (if only these ideas would become commonplace!).

First, Debian (a FOSS operating system which is the root of Ubuntu, Knoppix, Xandros and many other Linux distributions) requires that their official packages, a collection of software prepared for easy administration, must adhere to a very mature policy. Debian’s policy is a marvel in the FOSS world and to a very large degree is responsible for its strong support for both customization and upgradeability. I think Debian’s reputation for stability and maintainability is almost certainly due to their decision to develop a consensus-driven policy that its software must implement.

For example, the Debian package maintainer, Luigi Gangitano, for Drupal, a FOSS content management platform, did a great job making the software both customizable *and* maintainable. The package supports configuration of multiple virtual hosts which can all be upgraded at once! And the Debian drupal6 package stores the look-n-feel in `/etc/drupal/6/themes/` so that each site’s GUI can be customized without interfering with upgrades. If only all web applications were built to be as maintainable as Debian’s Drupal package!

Another example is the *overlay* support included in RT: Request Tracker, a FOSS ticket tracking system. This allows putting replacement subroutines in special files in `/usr/local/share/` which overlay or substitute the upstream code. This approach is more likely to break on upgrades, but it supports minimal changes to the business logic with a decent chance that upgrades will be smooth.

There are countless more examples from the FOSS world of innovative solutions to inter-accommodate customization and upgrades in support of **eternally regenerative software administration**. What are some of your favorite examples?

Posted by CJ Fearnley *in* Debian, Eternally Regenerative Software Administration, *1 comment*