[Cb:Devel] Extreme Systems Administration

Ken MacLeod devel@casbah.org
10 Jan 2001 13:19:10 -0600

- Next message: [Cb:Devel] Orchard/C goes Alpha!
- **Messages sorted by:** [date] [thread] [subject] [author]

[Cc'd from a post I sent to c.s.xp.]

We're getting ready to deliver several packages into a new production environment. I was wondering how others apply XP values, principles, and practices into operations/sysadmin.

I have an evenly split background between development and sysadmin, and was very heavy on the admin side for seven years up until two years ago. I've developed several practices and rules that I believe share many XP values and principles and mesh well with existing XP practices.

SysAdmin development is still development

> When developing code by or for SAs or production systems, you're developing code. All Extreme Programming practices apply.

Maintain minimal-delta systems

> Production systems should have the bare minimum of differences from the vendor's base install. They should be summarizable on one sheet of paper, the manifest. A manifest should list the base install, the add-on packages installed, the files used from source control for configuration, and any remaining hand-made changes. Per-system manual changes should be kept to a strict minimum. Use automated installs whereeever possible.

what format?
Stored where?

Refactor mercilessly (applied to systems)

> Manual changes are migrated to source control, common configuration and implementation is migrated into releasable packages. Automate manual procedures. Goals: keep the system manifest simple and keep "work effort" off production systems.

could we make
a tempus config
pkg? so BCS build
patch + pkg =
tempus machine?

Use production manifests to build test systems

> Test systems should be rebuilt regularly (every or every few iterations) using production manifests. This "tests" the production manifest and keeps the test systems in-sync with the production systems. Use test systems to aid in refactoring production manifests. Use automated installs to make rebuilding test systems faster and easier.

Keep logs of
updates & patches
what format?
where?

Keep systems up-to-date

> Apply security and vendor updates regularly. Give updates a chance to "settle" before applying them. Use test systems to "age" updates before applying to production systems. Keep up with major product and OS updates, they're easier when they're gradual. Use test systems to do parallel development, updating a test system with a new OS/product and identifying any issues with existing systems.

Maintain operations procedures

Develop and factor common operations and maintenance procedures, as checklists or outlines. For less-often used procedures, note the last time it was used. Double-check unfamiliar or older procedures before they're performed. The default procedure (typically used only in emergencies) is to log every step you do.

Pair operations

All access to production systems, outside of routine backup or job-control, is performed as pairs. Avoid using significant privileges (Unix's root) unless absolutely necessary. Factor out and wrap priveleged operations whenever possible (Unix's sudo).

Keep access to production systems to a minimum

At any given time, a production system should work indefinitely without intervention or support. Access to production systems should always be for a documented procedure (operations or maintenance). Enable and extend remote status tools to provide production status without requiring access. Use test systems and source control for checking how things are installed or working.

Use operational tests

Create (or require) operational tests that verify that a system, product, or package is working as expected. Both on-line (non-destructive, accessibility) and off-line (thorough) testing should be used.

Seperate production and development database activities

Database schemas and updates are delivered with packages. In-place database updates are repeatably testable and verifiable on test systems, prior to release on production systems. Tuning information is maintained across updates (either through feedback to source code or externally merged).

Keep the developer, test, production loop small

The operations group is a customer of development. Administrators are co-developers for many tasks. Every iteration release should be installed on a test system. Refactor installation processes and keep as simple as possible. Minimal production support is a coding standard.

Use package management extensively

Require vendors and off-site developers to deliver using your system's packaging tools. Package any products downloaded or developed locally. Merge common, infrequently changed packages into your auto-install or vendor media.

Parcel your time

Prioritize operations and maintenance over development, refactoring over new development. Spread non-weekly tasks so that several occur every week. Keep a record of time spent on tasks for future scheduling. Group related operations together, and complete them as a whole (rather than individual ticket-tracking). Rotate "on-call" as a daily or weekly task, so that other's tasks can be completed uninterrupted (ie. avoid mixing on-call with other tasks). "Tasks" are the primary internal unit of staff load, with respect to team velocity.

Communicate schedules and policies

Keep your schedules and policies in an open place (web page or bulletin board). Let customers (internal users, stakeholders, and developers) know where you're spending your time. Let them know which tasks are fast-tracked (common operations) and which require prioritization and queuing (development), let them know what your queue is. Let customers prioritize queued tasks among themselves.

Test disaster recovery by swapping in new production systems

A test system built according to the manifest of a production system needs only the production machine's data to replace that system. Do that at least once a year for every production system.

Keep systems modular

Develop sub-systems (packages, databases, resources, configuration) so that they can be moved from one system to another easily. Migrating to a new version of a system is more easily done one sub-system at a time.

Migrating to Extreme Systems Administration

Seperate your work effort into "old style" and "new style." Begin by bringing in new systems in the "new style." Migrate sub-systems to the new systems one at a time, refactoring them in the new style when doing so. Pay particular attention to simplification and refactoring at this point, it will pay off greatly. Rotate staff through old-style and new-style efforts until the new-style is completely in place -- don't share tasks between old-style and new-style. Take it slowly, generally no more than one or two sub-systems per iteration, for least interruption and greatest reflection time.

I never really had the framework (planning, management, communication, feedback) for these practices until I was introduced to XP. The two mesh so incredibly well. I believe this is an excellent start for a page on the XP Wiki if people are interested. I'd like to hear from others what their experience has been.

-- Ken

- Next message: [Cb:Devel] Orchard/C goes Alpha!
- **Messages sorted by:** [date] [thread] [subject] [author]