



A structured approach to building software.

By **Christopher Berarducci**

The original founders of Palm Computing founded Handspring in July 1998. This core team had stayed together through a variety of acquisitions, initially by US Robotics and 3Com. A number of the engineers from Palm joined Handspring when it was founded and, with them, brought many of the development tools that they had favoured at Palm.

In September of its second year, the company had just delivered its first generation of organisers to the market. There was a development process in place, but there was room for more automation and stability to be built into the development process.

The first major change that occurred was moving the software configuration management (scm) system from a Windows server and onto a dedicated Solaris server. The main reason for this was the Windows server was not dedicated and the system resources were often fully utilised. During this process, the team implemented journaling on a separate disk, had an uninterrupted power supply installed and implemented a system of doing backups every night.

That all ran smoothly until something went wrong. Multiple failures with the Solaris hardware occurred and, to this day, it is still not clear what the root cause of the issues was. While the downtime was limited to a few days, it was learnt that an onsite hardware backup was needed. This was achieved using a second Solaris box

similarly configured. The disks were on an external RAID array and able to be moved from one cpu to another (the RAID array was mirrored and hot swappable).

Since then, the scm system has proved to be very stable and has delivered solid performance. In the event of another

hardware downtime it is expected it would be limited to 24 hours or less.

Configuration

The in house build system is called ASH, which stands for automated system harness. ASH provides the frame-



SOLID FOUNDATIONS



work for tasks to be defined, requested (as in the case of a build), accepted or rejected, posted and for notifications to be sent to the relevant people. Software defects are tracked in 'Tracker' – bug tracking software that was originally derived from Bugzilla.

The scm system, Perforce, is tightly integrated with the development team, Tracker and ASH. Most developers are working with Cygwin on Windows but there is some Macintosh and Linux work being done. All of the tools are supported across these platforms. The team uses Perforce change numbers in its build scripts to ensure the correct source is used for builds and in turn move existing bugs to verify in Tracker.

To enhance both the explicit and implicit collaboration between development teams, two scripts focused on automating merging have been implemented.

The first script automates the merging of a single change number. Only the changes submitted with the original

change number are merged into the destination branch as a single change number. To understand this, it makes sense to explain a little more about the architecture of the scm system.

Perforce uses changelists. A changelist is a list of files, their revision numbers and operations to be performed on these files. Commands such as add filenames and edit filenames include the affected files in a changelist. When a changelist is submitted to the depot, the depot is updated atomically: either all of the files in the changelist are updated in the depot, or none of them are. This grouping of files as a single unit guarantees that code alterations spanning multiple files are updated in the depot simultaneously.

From a merging point of view, what the scm system might normally be con-



figured to do is merge one entire branch (say branch a) to another branch (branch b) at set time intervals. If, for example, ten changes had been made across five changelists in branch a, these

changes would be merged into one changelist in branch b. While this is a very valuable feature, we sometimes want to preserve the comments and the grouping of changes in the destination branch.

The second script, actually a set of scripts, is based on the concept of a change review daemon. When a source change is submitted, the comments and source files are parsed and a copy of the information is stored in ASH, within an sql database. Comments that contain specific keywords, like 'fixed bug 1234' result in those check in notes also being added to Tracker, the software defect system.

All changes relating to a bug are related to a single change number in the scm, which allows those changes to be automatically integrated throughout the tree. For example, if a bug is fixed in the release branch, then the associated changes can automatically be merged back to the main line and all the other

relevant branches.

The daemons also add a link from Tracker to the relevant change number in Perforce and can be viewed dynamically via the P4Web web browser interface. This allows engineers to go straight to the relevant files and to see all the associated change history.

A comment that contains a keyword like 'fixed bug 1234' might also result in a state change in Tracker occurring, where the state would move from 'open' to 'in build'. When files are submitted, ASH also associates them with any projects that build in that branch and if configured would move the bug to 'verify'.

The main reason to store a copy of the SCM meta data in the sql database is to provide optimal performance for the engineers while still providing a flexible way to produce web pages and reports.

This only becomes an issue when you have heavy automated processes going on and, of course, is dependent on the hardware configuration.

When an engineer needs a build, they visit a web page and request the build, a default change number is used (the highest one) or a specific change number can be specified. The build is assigned to one or more cpus to be compiled and the results are posted to a release server.

Quality assurance, via another web page, then has the option to automatically accept the build for wider distribution or to review it and then accept or reject it for wider distribution.

Overall, the automation of these systems has been a success. We have a stable, flexible development platform that saves time, money and effort. Along with the hardware set up and operating system changes, keys to this success are the automation scripts that work with the underlying architecture of the scm, with its change numbers and how these relate to the source tree. **NE**

Author profile:

Christopher Berarducci is a software process and automation architect at PalmOne, formerly known as Handspring. PalmOne is the manufacturer of the Treo.

"We have a stable, flexible development platform that saves time, money and effort."