www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

1. Introduction

At SPK and Associates we are often asked for direction with respect to versioning control systems. The management of software and related artifacts is core to your development process and a key factor in determining the operational productivity of your organization. But where do you start? There are dozens of configuration management systems available to choose from. Some are free, some are pricey. Some are integrated with defect tracking and project management capabilities. Others are stand-alone tools focusing only on version control. The following describes one of the projects I went through with a local software company. And upfront I'll state that there is no one right answer. Hopefully reviewing the process may help you down the path of determining: "Which software configuration management (SCM) system is right for you?"

2. Setting the Stage

This company was an established software vendor. Their target organization consisted of 300 engineers distributed between India, Europe and the U.S. Their existing SCM systems was a mixture of primarily Concurrent Versioning System (CVS), and then some RCS, SCCS and ClearCase. There were extensive wrapper scripts tied into their build, test, eco and release system.

3. Establishment of Requirements

After a series of discussions with development line staff and management we were able to converge on some high level requirements.

- Scalability & Performance

The tool must scale to handle a large development organization with hundreds of developers world-wide. It should be excellent for distributed R&D interaction.

Performance must be equal to or better than the CVS environment.

- Branching/Merging

Branching and Merging must be extremely easy for the line R&D Engineer to execute.

The tool must be able to track and report on all branches and merge history. Graphical depictions must be available.

Built-in and robust conflict resolution tools must be available.

- Ease of Use

All users (Developers, PV Engineers, Tech Pub, PM, and CM engineers) must be able to quickly use the system without extensive training. It should be intuitive.

- Administration Cost

The system should not require any additional CM or IT dedicated people

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

resources. High-availability and a robust backup strategy must be included. The system should not be a huge drain on the network.

- Conversion/Deployment

The system should not require any re-architecture of software product (cutting it up for repositories).

All SCM file history should be able to be carried forward.

The system needs to have a command line interface as well as API's for inclusion of other tools/processes/IDEs.

- Support

The system should come with excellent support as well as a track record of success with similar sized ISVs.

4. Narrowing Down Number of Systems

There are two major camps that most of the SCM systems fall into:

1) Vendors with solutions for application life-cycle management (ALM).

2) Open source projects and vendors focused solely on SCM

Vendors in the 1st camp are strongest on process and change management, often integrating other capabilities such as defect tracking and project workflow into a solution offering. IBM/Rational's ClearCase/UCM and Team Concert led the pack in this area with other vendors such as MKS Integrity Platform and Microsoft's Team Foundation showing comparable maturity and features.

Camp 2 vendors focused primarily on one thing – Configuration Management. Our shortlist vendors included Subversion, Perforce, GIT and Accurev.

The customer decided that they would like to have a single purpose point tool that would fit into their portfolio of applications as opposed to a solution platform. Thus the camp 2 vendors moved forward in the evaluation.

GIT was later dropped to the perception that:

- It not being suitable for something extremely large or complex. The customer's product hierarchy was 19 million lines of code. The speed and efficiency gains largely come from using smaller, distributed repositories. GIT users would each have a full copy of the source code.

- It being based on a distributed source code environment (no central repository), where each person essentially has their own branch. This was thought to be potentially confusing to manage given there may be hundreds of

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

branches to merge in with no clear master.

Accurev (an early favorite with its powerful stream-based architecture) also lost traction with development community and was removed from the list. As most of the users had primarily a CVS back ground, they found the stream model too much of a paradigm shift.

Thus we had now had shootout between Subversion and Perforce.

5. Test Team and Sites

We began the process by requesting volunteers from R&D and CM worldwide. The responders became input providers to the evaluation.

We ended up with a team of over 30 people including R&D, Product Validation, Program Management and CM engineers.

Sites Represented: U.S. West Coast Site, U.S. East Coast Site, India Site, Scotland Site

6. Perforce and Subversion Evaluation Environment

The CM team created a full product installation for both Perforce and Subversion. Each was a complete migration of the existing product CVS repository data (all files and history).

- CM conducted high volume performance bashing.
- CM created how-to scripts (focusing on branching/merging) and sent them to each of the testers.
- The testers were given approximately 3 weeks per tool to play with and to provide feedback.

6. Scalability & Performance Results

SITE	CVS (minutes)	Subversion (minutes)	Perforce (minutes)
West Coast	30.13	44.28	**16.27**
India	**20.40**	27.20	36.34
East Coast	35.52	1.06.43	**13.05**
Scotland	1.38.08	1.15.04	**6.58**

Table 1 - Comparative Average Run Times

A full checkout of all the source code by CM was done at each site multiple times. The average run times show that Perforce was the fastest overall (with the exception of India). The India numbers we believe were skewed by a network/tuning issue and would eventually track similarly to other non-U.S. sites.

*note: The 1st time the command is run in Perforce there is one time setup cost. Numbers presented here are the on-going user experience using the proxy setup.

Both Subversion and Perforce scale up to handle thousands of users. With respect to distributed development, Perforce had the edge through its implementation of

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

proxy caching, which caches and serves files to users in remote locations. This helps reduce WAN traffic. The same synchronization could be created with Subversion, but it would require additional customization scripts.

7. Branching/Merging Results

Branching and merging were simple using both Perforce and Subversion. However, the feedback received from the team was that Perforce is superior to Subversion.

With Subversion 1.5 the concept of merge tracking was introduced. While it did improve its otherwise earlier manual process of keeping track of your changes, it still fell short of Perforce. Perforce was a more complete solution with respect to tracking and reporting all the history and merge data. It was also easier to merge file changes across multiple branches automatically.

Perforce also had a built-in superior conflict resolution tool.

8. Ease of Use Results

Subversion is similar to CVS with respect to command line language. CM users found it easy to use due to familiarity. However, Perforce was the overall winner due to its rich native GUI offering. For example, the revision graph tool displays a detailed branching history of each file for easy visualization of code propagation. The time-lapse view provides a graphical view of the complete content history of an individual file across branches.

Overall, via its command line or its rich native GUI it was very easy to conduct Branching/Merging, Conflict Resolution, and general CM operations in Perforce out of the box.

9. Administration Cost

Both Perforce and Subversion have light hardware and people resource requirements. However, Perforce provided a better ROI due to an increased number of built-in capabilities. Many of the same capabilities could be recreated by scripts in Subversion, but that would require additional development and support resources.

As an example, I already mentioned the Perforce WAN advantages due to the proxy cache. Both systems allow for a SAN roll-over backup strategy.

10. Conversion and Deployment

Converting the product codebase to Perforce would take less engineering effort then moving to Subversion.

CM creation of the evaluation depot was done in a few hours. The equivalent Subversion repository took days (and there were errors to resolve). Both tools did not require any re-architecture of the product hierarchy.

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

There would be less new scripts required with a Perforce conversion due to its more complete feature offering.

Both tools offer robust APIs and plug-ins for popular IDEs (Eclipse).

11. Support

Perforce support is purchased as part of the license agreement. During the evaluation we had a few questions and issues with Perforce. They were very quickly responded to and resolved by Perforce support. Subversion is an opensource tool. General questions can be posted to the community forum or formal technical support can be purchased through Collabnet.

Both tools are used worldwide – More use Subversion as it is free (especially colleges and universities). Some companies which use Perforce included Google, Nvidia, Synopsis, Fujitsu, Hitachi, Philips, Samsung, Silicon Graphics, and Siemens.

12. Software Cost

Subversion is opensource software and is downloadable for free. Support can be purchased from Collabnet. There is a sliding scale depending on desired service level, up to platinum level at ~$30K per year.

Perforce pricing is variable based on number of users. For the entire business unit (at 300 Users) we were looking at around $218K for year one, with and annual support cost starting year 2 of around $48K/year.

13. Overall Rating Card

Attribute	Subversion	Perforce
Scalability & Performance		**Clear Winner**
Branching/Merging		**Clear Winner**
Ease Of Use		**Winner**
Administration Cost		**Slightly Better**
Conversion & Deployment		**Winner**
Support	**Tie**	**Tie**
Software License Cost	**Winner**	

Table 2 – Overall Rating

14. Summary Statement

Both Subversion and Perforce represented a significant improvement to core functionality and performance over CVS (where most of the products originated). With the need to migrate product hierarchy (and later other parts of the company) to a more agile development model, the team believed Perforce provided the overall best versioning system solution given the requirements.

Ultimately Perforce was successfully deployed to the 300 users and began getting traction in other business units as well.

Carlos Almeida
SPK and Associates
Architect, Software Engineering