Free source code management tools are becoming more and more popular. But is free just as good as commercial offerings? Back in the 1970s, software configuration management (SCM) meant version control. Anything more than that was an in-house, advanced solution. This situation persisted through the 1980s. Two operating systems—Digital Equipment Corporation's VMS and Apollo Computer's Domain platform, a workstation variant of Unix—got into the act, providing various levels of version control. There were a few impressive proprietary, homegrown solutions, but they were largely invisible to the overall software industry.

Finally the 1990s arrived. Yes, there were still version control tools—RCS, CVS, and PVCS, to name a few—but there were also tools such as ClearCase, Continuus (which later became Synergy), STS (later CM+), MKS, and other evolving commercial tools. There was strong competition among the vendors. The UK research firm OVUM performed annual SCM tool product reviews that were highly respected and anticipated. Vendors had to improve their tools in order to stay competitive. Suddenly, the computing world was introduced to real SCM solutions, focusing on the broader software development lifecycle and on process and automation.

However, toward the end of the 1990s and into the new millennium, advances in SCM slowed and mergers took place, such as IBM's acquisition of Telelogic (Synergy) and Rational (ClearCase). The SCM industry continued to advance, but with reduced competition among vendors, the focus shifted more to lower administration costs than to investing in new features.

There was a move to glue together parts of a solution: requirements management, version control, change and configuration management, build control, test case management, document management, and even problem tracking.

Some integrated solutions knitted together two or three of these, others combined the tools into a comprehensive suite, and some provided many parts of the solution in a single integrated tool. With few exceptions, these solutions came with hefty price tags. The industry slowly adopted the term application lifecycle management (ALM), synonymous with computer hardware's product lifecycle management.

There were still new CM tool startups, including Accurev, and Microsoft's VSS—but, in my opinion, these were largely version control tools with a new twist here or there.

Subversion and Git

More recently, Microsoft's TFS and IBM's RTC have shown some real advances. But the software industry has embraced newer version control tools, with Subversion and Git topping the list. Why?

To put this in context, Git and Subversion, both open source version control tools, are battling it out for dominance in the SCM industry, and many organizations are regressing from stronger SCM solutions to more basic Subversion or Git. Some commercial SCM tool vendors have reacted to integrate their tools with these open source version control solutions.

Software teams need advanced SCM or ALM solutions with real benefits that provide real productivity to all product team members. These benefits include fail-safe reliability and accessibility, near-zero administration, full change package support, a mature SCM process, easy process customization (rather than process buried in scripts), advanced user interfaces, reduced training requirements, comprehensive SCM metrics, generation of required SCM and release documents, data security, and navigation of traceability relationships.

With today's SCM technology, it's possible for users in each role to increase their productivity and for the entire product team to have all required SCM information at their fingertips. Good SCM tools should result in higher quality products with lower CM costs than basic version control tools.

Everyone Is Transitioning to Simpler SCM

I believe there are eight reasons SCM is reverting to version control through the use of Subversion and Git. But are these reasons really justified?

1. Price of commercial tools

The reality—or perception—is that commercial tools are always expensive. ClearCase, the dominant commercial solution just a few years ago, carried a price tag ranging from thousands to tens of thousands of dollars per user. IBM's acquisition set the stage for costs holding fairly steady. And even though there are reasonably priced, highly capable commercial SCM tools, the perception is that the capabilities doesn't justify the added expense.

In addition, commercial SCM tools typically consume much customization effort, requiring consulting and training, although some of the new SCM and ALM tools have driven those costs down significantly through easy out-of-the-box installation and simpler configuration procedures.

However, building processes around free version control tools will cost significant resources. And from a training perspective, the biggest cost is in lost salaries. As a result, a mature process and easy-to-use technology are both needed to reduce training costs. These are somewhat lacking in open source solutions and in many commercial tools, too.

2. Benefits of open source tools

Open source tools are inexpensive to acquire and maintain. The source code is in the public domain, so there is no chance of the vendor going bankrupt, and with so many contributors and experts out there, the tools should continue to receive support and enhancements.

However, it is difficult to make architectural changes to an open source product, as this is disruptive to the community and knowledge base. It also requires dedicated resources to see such changes through, which can take a significantly long time to develop.

Speed of updates can also be a problem due to a gradual response to market demand. In contrast, commercial providers pride themselves on rapid customer response for new features and capabilities.

3. Integration of commercial offerings bundled on top of open source tools

In the commercial SCM and ALM market you'll find commercial vendors selling Subversion and Git solutions. Version control is a very visible component of an ALM solution, and when a vendor bundles one of them into its solution, that appeals to a market that has already adopted one or the other. On the other hand, a free tool bundled in a commercial package results in the perception of the loss of the free benefit.

4. Startups without the experience of full configuration management capabilities

Developers don't like administration, and if you put a group of unseasoned developers together, the last thing they want is to put some SCM administration in place. This is actually a selling feature for open source solutions. There is no need to contact vendors or evaluate solutions—just download what everyone else is using. You can find the minimum feature set you need right now.

Experienced developers, on the other hand, recognize the benefits of full ALM solutions. They know it's best to start out with all the capabilities at hand, especially if that solution increases developer productivity.

5. Marginal benefits of commercial offerings

Just as the cost of some commercial tools can establish the perception that commercial tools are expensive, the functionality of some tools can paint the perception that commercial tools are only marginally better than open source counterparts. And in some cases, that's true. So why pay?

In my experience, there are several commercial tools out there that will pay for themselves within a few short months and then continue to accrue benefits. It doesn't take a lot of marginal benefit to cover the license costs of a commercial tool.

6. A lack of customization capabilities in commercial offerings

Open source version control tools have very limited customization capabilities, including scripts, triggers, and settings—perhaps sufficient, considering version control is a small part of the SCM and ALM puzzle. SCM and ALM tools, on the other hand, must support a greater variety of users, process, and data. Whereas version control needs may slightly differ between one organization and the next, this is not the case for SCM and ALM.

And while some commercial tools support large process variations that fit many projects, other offerings are much less configurable. SCM and ALM tools need to support significant customization and configuration, including the definition of metadata, tuning of the user interface for specific roles, defining the presentation and navigation of data, defining custom information links to to-do lists, and modification of process. In addition, the tool should provide documentation support, report and dashboard creation, and metrics required for a project.

With a high level of customization capability, each user can look at the complexity of SCM and ALM through views specific to his roles and requirements. The easier to customize, the more value the tool adds, resulting in increased productivity.

7. An overall poor understanding and poor marketing of the true benefits of full ALM

There are plenty of inexperienced team members out there. But they are going to remain inexperienced if the benefits of a full ALM solution are not easily and readily explained. The software SCM industry has not done a good job of educating the industry or marketing ALM.

Proper marketing of true benefits might take the form of annual tool competitions, where real-world SCM and ALM issues are addressed by all commercial tool suppliers, and even open source solutions.

SCM product reviews can help, but the complexity of SCM may preclude a thorough review and result in comparing only the basic common elements of each tool. You cannot compare an open source tool such as Git to an advanced, modern SCM and ALM tool. It would be like comparing a bicycle to an automobile.

8. The perception that building around open source tools is easier to sell to management than capital expenditures of ALM tools

"How much does the tool cost?" is usually the first question. And if the answer is that it is free because it's an open source tool, then the response a software manager will most likely give is "Great! No cost? Go for it!" However, a decision like this would never pass a business case review. The cost of licenses is not the largest cost of SCM. Training, process implementation, scalability, and integration with existing systems can be very costly.

Every company needs an ALM solution. How much of that is manual or done piecemeal is a separate question, but the cost of ALM is the cost that has to be measured in a business case.

Free version control, no matter how good, is not an ALM solution. Solutions may be built around it and engineered for cost-effectiveness, but it's even better to have a version control component specific to a full ALM solution. Then certain things become more obvious: You don't check in files; you only check in changes. You don't just type in comments; you reference and link to approved problems and feature activities.

Advance!

SCM technology appears to be regressing. It's customer-driven. But look again and perhaps you'll see that this shouldn't be the case. The software industry, in its adoption of version control instead of SCM, is charting a course that is not much different from what we witnessed back in the 1980s.

What do you do to fill in the holes in your SCM? Or is version control your only third-party tool? How do you justify the cost of maintaining your own scripts to support the version control tool you use? Are you considering changes to how you perform SCM or version control? Take a good look at what's available out there. It's time to advance! **{end}**

farah@neuma.com