Software Configuration Management Helps Solve Year 2000 Change Integration Obstacles

Tom Burton
InRoads Technology

While defense organizations implement year 2000 (Y2K) changes, they continue to implement many day-to-day changes on these same applications. These two activities are usually undertaken separately, resulting in the potential for disjointed development efforts and software crises. Successful organizations will need to manage Y2K conversion and other changes through effective software configuration management.

The defense software engineering community is grappling with the challenge of changing code in mission-critical applications so that they will work properly in the next century. This conversion saga requires the implementation of thousands of changes to application areas that have not been touched for years.

It is laborious and time consuming to find areas where code must be changed. Engineers, for instance, must wrestle with missing source code, wrong versions, redundant modules, and sloppy documentation. They also must deal with the thousands of additional fixes that have to be implemented throughout the conversion project. It is a constant worry that unwanted changes will fall unnoticed between the cracks.

Yet, while date-change activities are taking place, software engineers must conduct day-to-day revisions to the same applications. Such modifications may be implemented to remove bugs and add new features, functionality, or enhancements to these applications.

The scope of these day-to-day changes also is substantial; the code of mission-critical applications is constantly being modified for various reasons. For instance, different versions of defense applications often must be built to accommodate a multitude of simulation scenarios, e.g., it is much cheaper to test a missile's software under various simulation scenarios than to conduct actual launches for the same purpose.

Software engineers must constantly modify an application as it goes through various lifecycle stages. Additional challenges include updating documentation while the changes are in progress, making changes to firmware, and factoring hardware changes into the equation. None of these revisions are trivial given the complexity of today's mission-critical applications. If not properly managed, these changes may cause the system to collapse.

Often, the problem is that the conversion process and the day-to-day application changes are treated as separate projects, and therefore implemented by multiple teams without effective coordination. The development environments in such organizations could face a crisis of gigantic proportion with the turn of the new century.

The Y2K conversion project and day-to-day changes must be conducted harmoniously, since work may involve thousands of changes to millions of lines of code. Otherwise, changes may be accidentally left out of the end-user's version—an unacceptable mistake. Lives can be lost if, for instance, an automated combat response system triggers an attack against a friendly ship or aircraft. Developers cannot afford to deliver software that is only *mostly* the right version.

The Software Configuration Management Difference

Fortunately, development organizations can potentially deliver 100-percent-right software by placing Y2K conversion and day-to-day development activities under the common umbrella of software configuration management (SCM), also commonly referred to as process configuration management. SCM strictly organizes the tasks and activities that maintain the integrity of software product configurations, ensuring configurations are correct, i.e., that engineers are working on the right source code and the right versions of an application.

Many development organizations, whether they know it, spend 25 percent or more of their time trying to manage their configurations. The good news is that the defense industry is ahead of the pack for having recognized the need for automated SCM long before it became an established concept in the commercial sector. The bad news is that—although the importance of SCM has been recognized for years and millions of dollars have been spent on SCM tools—few software development organizations do a great job of SCM.

In fact, most average software development projects are only able to keep 75 percent to 80 percent correct software configurations. Many are lucky to maintain a 50 percent to 60 percent correctness, and quite a few projects have software product configurations less than 50 percent correct [1]. We should not expect better results for Y2K conversion projects unless we implement better controls than the current state of the practice. Considering the critical need for Y2K conversions to be precise, there is an obvious need for most organizations to become much better at SCM.

Good SCM

Good SCM occurs when the configurations are continually 100 percent correct—there are no lost or missing changes, all the correct software components and versions are included in the

builds, and no changes targeted for the next release somehow end up in the current release. Good SCM also provides proof that the configurations are 100 percent correct at any time in the development lifecycle.

In sum, good SCM is achieved when the software product configurations are 100 percent correct and contain all the wanted changes, when the development organization is 100 percent certain that these applications are complete, and when the development team can demonstrate that these applications are 100 percent correct.

But how does an organization get to that point? To be effective, the Y2K conversion project and the day-to-day development activities must be implemented concurrently and in parallel under the umbrella of good SCM. This means that software engineers must take all the changes from the Y2K projects and all the changes from the day-to-day projects and integrate them together, test them, merge them, and not leave anything behind.

SCM Pitfalls to Avoid

It is difficult to achieve good SCM without glitches. Three main issues significantly impact the ability of the organization to establish SCM.

First, poor SCM often results in changes being made, but then not put back into the product configuration. This happens when programmers forget that they have made a particular change, then the developer reinserts the wrong version of the program. This typically occurs when the programmer does not understand or follow the organization's established SCM processes.

A second problem can occur when parallel and concurrent activities are taking place: one developer's changes to a version overwrite another developer's revisions to the same version. This is known as "change regression," and though common, this problem is often overlooked. Such a scenario typically occurs when there is no automated process CM tool in place that can track all changes to the system, allowing developers to compare their changes and select the ones they want to keep.

A third SCM problem occurs during the build process, when source programs are compiled and linked to create executable programs. Because the build process is typically automated, wrong versions can easily be incorporated automatically into the executable program. This happens when developers do not know from where the build is picking the executable program; therefore, they create and test incorrect executables. Many additional SCM issues also come into play during the build process, all of which impact the integrity of the software product configurations.

Implementing Good SCM Practices

Three critical practices can prevent these pitfalls and promote good SCM, helping achieve the goal of 100 percent correct mission-critical applications to users.

- The organization must formally document its development processes and use these processes as a road map to effectively integrate the Y2K projects and the day-to-day changes to these applications.
- The team must be educated as to what constitutes good SCM practices. Training should be viewed as a process that spans the development lifecycle. For instance, training can teach the team the organization's processes, help shorten the evaluation of SCM tools, help deploy SCM tools, and facilitate implementation. Many outside resources can help achieve SCM competency. Some vendors even offer ready-made SCM classroom materials and multimedia computer-based courses, which can effectively be used for in-house training.
- The development organization would do well to rely on leading-edge process configuration management tools, which help automate, distribute, and merge the changes associated with the Y2K project and day-to-day development activities. These client-server solutions provide facilities to map organizational and development processes into the tool. They also provide a complete audit trail of all development activities so that the development environment can guarantee that the applications delivered to customers are 100 percent complete.

Conclusion

Organizations that treat Y2K conversion projects separately from day-to-day development activities are likely to experience significant trouble and setbacks. To start on a path toward 100-percent-correct software, these organizations will have to put these two efforts under the common umbrella of SCM. Achieving good SCM is possible in this framework when software processes guide the overall development effort, when developers are educated about effective SCM practices, and when these efforts are automated via a leading-edge process CM tool. This will put the software development environment on a more secure path to success. ◆

About the Author

Tom Burton is CEO of InRoads Technology, which specializes in providing tools and education for the successful implementation of software configuration management. He has more than 10 years experience in software configuration management.

Voice: 805-967-4545
Fax: 805-964-4790
E-mail: tburton@inroadstech.com
Internet: http://www.inroadstech.com

References

1. "Configuration Management Industry Outlook," *InRoads Technology*, January 1997.
2. "The Year 2000 Digit Crisis Sounds the Alarm for Active Control of Software via Process Configuration Management," white paper by Tani Haque, CEO of SQL Software, November 1996, info@sql.com.
3. "Creating a Culture for Successful Process Configuration Management," white paper by Tani Haque, CEO of SQL Software, July 1997, info@sql.com.