“Global Software Change Management for PVCS Version Manager”

IKANALM
ALM through evolution

www.ikanalm.com

Summary

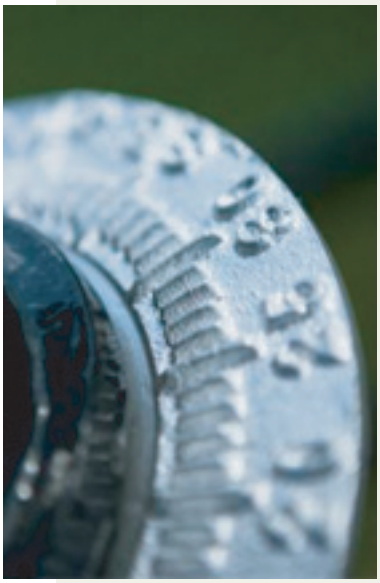
PVCS Version Manager is considered as one of the leading versioning tools that offers complete versioning control. But when building, deploying and tracking the life-cycle of your software applications, you can lose sight of your code as PVCS Version Manager typically does not cover these areas. Many PVCS Version Manager users would therefore like to evolve to this next level, but they want to keep using their favorite versioning tool. The integration between PVCS Version Manager and IKAN ALM solves this problem by offering a unique and flexible process-centric software change management solution for both local and distributed development teams, keeping the impact on and the tool-specific training needs of the software engineering team to a minimum. IKAN ALM continuous where PVCS Version Manager stops and helps project teams to further automate the complete software lifecycle management process, combining both continuous integration and life-cycle management, offering a single point of control and delivering support for your build, deploy, release and software life-cycle management and approval processes.

IKAN
DEVELOPMENT

Introduction

Software development, seen in a historical perspective, is a very young discipline. In the early stages, software development was seen as an art (the developer is always right, even if he was wrong, and beauty won over effectiveness and efficiency), as it was seen as supporting business objectives. In the recent decades however, software development started to organize itself: initially to make life easier for itself, finally to answer internal and external driven productivity and compliancy needs.
Building a software solution involves many complex processes, roles and deliverables, which need to be managed to fit together. Streamlining these processes is a major effort, particularly when there are many people involved. A truly comprehensive software configuration management (SCM) solution manages not just the simple versioning of your source code files it also facilitates support for Continuous Integration, accurate and predictable build management, provides the ability to deploy the end result as well as offering approval processes and manages the complex run-time dependencies of many applications today.

What does SCM comprise?



Although most developers are happy with just version management, in today's world where users have become heavily involved users, production people, auditors and compliance officers and finally shareholders, compounded by the global context in which we operate, we are faced with new challenges: offshore, virtual or outsourced development to name just a few. We now need to answer questions like who did what, where and when, and who has control, what are the fall back options, and how to comply with the new laws on corporate governance. These new requirements demand that SCM (Software Change Management or version management in the narrower sense) not only keeps versions of software code, but that it also offers developers independent and controlled build management, deployment, lifecycle management and approval mechanisms.
Every role within the organization has a different area of interest. A developer wants to have early feedback on the code he committed in the trunk stream (Continuous Integration) and build the project in his IDE with the correct latest sources and common libraries. A project manager wants to have a clear overview of the project status: is the latest code in the trunk buildable, do the unit tests run successfully, what is in the QA phase, what is the current production version?
Production operators prefer an automated deployment process, where they can control the environment variables. Finally the CIO and CEO of a corporation want to see an automated and repeatable process with an audit trail.

	Developer	IT Management	Production	Management/audit
Why Versioning?	To keep track of the changes	Safe storage of all historic data	Easily revert to a prior version	No loss of data
Why implement a continuous integration process?	Concentrate on developing software Get early feedback on committed code	Early feedback on code quality Find weak spots	Get high quality production code	Fewer errors Repeatable process Faster and shorter release cycle
Why have an automated build?	No loss of valuable time trying to build manually	Allows to do more builds and gives rapid feedback	Everything is coordinated by a script	Prevents mistakes
Why approval management?	Improves communication across the project team	Control the evolution in the different stages of the life-cycle Build in audit moments	Control deployment to the production servers	Traceability Who authorized
Why have an automated deploy?	Guarantee that production will receive the quality code that I created	To speed up the process and help reduce errors	No manual intervention reduces risk	Increases the possible release cycle frequency and productivity
Why rollback process?	Have more time to fix defects	Can always revert to the latest good release	Refuse or eliminate risk of service outage	Ensure return to exact prior state Quickly resolve errors in production
Why life-cycle management?	No worries of building code for the test or production level	Have a clear view of the development process and status	Automate production deployment Reduces rework	Answers questions of Who, When, Why and What occurred?

SCM is process driven and is composed of several steps: it starts with versioning and finally ends with deployment. True Software Change Management therefore comprises the following aspects:

Versioning

The first step is versioning. Versioning sources are a best practice, which is considered evident for controlling the software development process. Whenever people startup a new software project, setting up a version control repository is the first step and is done even before the first code is created. When you ask developers what they need, they will answer you that a versioning tool like Serena® PVCS Version Manager perfectly supports their needs, as it versions their code. Today PVCS Version Manager is one of the most used versioning systems.

IKAN DEVELOPMENT

PVCS Version Manager interface

In general PVCS Version Manager is a commercial version control system. That is, PVCS Version Manager manages files and directories over time. A tree of files is placed into a central repository. The repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to recover older versions, or to examine the history of how data changed. PVCS Version Manager can access its repository across networks, which allows it to be used by people on different pc's. At some level, the ability for various people to modify and manage the same set of data from their respective locations fosters collaboration. Progress can occur more quickly without the constraints of having a single conduit through which all modifications must occur. And because the work is versioned, you need not fear that quality is the trade-off for replacing that conduit - if some incorrect change is made to the data, just undo that change.
PVCS Version Manager offers complete versioning control to its users, however it does not support all the other important phases in the software development process, such as build, deploy and life-cycle management, nor does it provide a traceable and auditable development process. That's where IKAN ALM comes into play. The IKAN ALM PVCS Version Manager interface seamlessly integrates PVCS Version Manager's easy-to-use PC version control within the IKAN ALM change management framework, by offering automation, control and visibility throughout the whole life-cycle, from managing a Continuous Integration build close to the developers, to automating the deployment into test, QA and production and organizing the related authentication and approval processes. IKAN ALM allows you to ad these next steps to PVCS Version Manager.

Automating Build Management

Build management orchestrates the complex software assembly, testing and packaging processes that go into producing the product executable. IKAN ALM ensures a controlled build process, giving you the possibility to reuse existing build/test or deploy scripts from leading scripting tools like Ant, Nant or Maven 2. It enables you to automate, accelerate and simplify software builds ensuring that your software applications get built the right way every time.

IKAN DEVELOPMENT

The IKAN ALM server monitors the PVCS Version Manager repository and jumps into action when it notices changes (commits). Then it will check-out a full or incremental load of the code base, run the build file and report on the results. What constitutes a build can be completely user-tailored: typically a build is a full compile of all source files, running unit tests and packaging the build result so that it can be deployed later in the life-cycle. Yet a build can also only consist on a packaging process of documents, or libraries (dll´s, jar´s, copybook´s, ..), or IKAN ALM allows the definition of an unlimited number of build environments within a level. Builds may be executed on a local or remote system, in other words "Distributed Builds" are supported. When a build fails, the responsible developers are notified and corrective actions can be taken. After a successful build, IKAN ALM tags the PVCR Version Manager repository with a unique build number to identify the sources that where involved in the build process: this guarantees "source to load" synchronization in every stage of the life-cycle. IKAN ALM also supports sources that do not need any source to executable transformation, like documents, include files etc.

Implementing Continuous Integration

Continuous Integration is an extreme Programming (XP) development practice where members of a team integrate their work frequently; usually each person commits their code changes at least once daily - leading to multiple integrations per day. Each integration is verified by an automated build (preferably also including unit tests and code checks) to detect integration errors as quickly as possible such as, for instance, a failing unit test. Early feedback on broken builds or failed tests is very important since such information will dramatically reduce the time to fix such errors, and, consequently, will also cut down on overall development time. It's obvious that it will take a lot more time fixing code that was written a week ago, compared to code that was just committed to the version control repository only minutes ago.

Getting the sources turned into a running system can often be a complicated process involving compilation, moving files around, loading schemas into database, and so on.It is clear that these tasks should be automated to run on a dedicated server, so that a repeatable build process can be started as soon as possible after the code-base in the versioning system has been changed.

IKAN ALM complements PVCS Version Manager with Continuous Integration support in different ways: it is possible to add a post-commit hook script in the PVCS Version Manager repository that will create a build request each time code is committed in a project controlled by IKAN ALM. Another way of implementing Continuous Integration is by setting up a schedule that will check after a predefined interval if there are changes in the project code-base of the PVCS Version Manager repository. IKAN ALM runs a central server which can be placed on any RDBMS. The server monitors the source code repository and jumps into action when it notices changes (commits). Its job is to check-out a full or incremental load of the code base whenever any part of it changes, run the build file and report on the results. What constitutes a build can be completely user-tailored; normally, a build is a full compile of all source files, running unit tests and creating the deployment archive.

Test & QA level

The development process does not end with the integration build. The result must be tested on a QA level, rebuilt with other compilers or operating systems, deployed under different application servers, and tested on distinct versions of the underlying application database. These are error-prone processes and thus ideal candidates for automation. This is the only way to guarantee that the code used to create the build result is the same as the code deployed and running in a test environment. During the test you should identify the problems that also might occur during production. It is therefore important that your test environment is similar to the production environment. If this is not the case, it's difficult to predict what will happen in production.

IKAN ALM provides the build/release manager with both a manually and/or automatically controlled build/release process for building/rebuilding and delivering applications independently of the logical and physical environment.

Simplifying deployment

Traditional change management systems have focused on activities close to the developer, such as source code control, but few mechanisms were in place to support software deployment activities. To be able to know which build version of the application must be retrieved from the build archive to deploy onto the designated target platform you need to automate the deployment of all your software assets. When you automate the prior steps, it's obvious that the last and most important step in the development life-cycle is also a controlled process: deploy to production. The production environment will probably resemble the test environments. However, you will not rebuild your application on the production server, and there will be more audit, notification and authorization involved before code is actually deployed.
IKAN ALM uniquely deploys applications throughout your enterprise: an unlimited number of deploy environments may be defined within a level, each identifying a physical machine, being a Windows server, Unix server or a Mainframe; a life-cycle may contain multiple logical steps, these are the Test or Production Levels.

Each deploy environment may be parameterized with dynamic deploy properties, which can be configured for allowing changes right before promotion to the next level. IKAN ALM integrates with intelligent scripting tools like Ant, NAnt and Maven 2. This makes it possible to solve complex deployment issues, like keeping life clustered servers, or synchronizing applications with their database. IKAN ALM knows which version of the application it must retrieve from its build archive and deploys it (in parallel if necessary) to the required target platform(s). IKAN ALM can also execute deployments to the DMZ (behind the firewall), using a number of secure communication protocols.

Streamline your approval process

To improve the communication within the project team it is advisable to set up approval and notification processes for this last step of the life-cycle. It's a good practice to notify project members that a delivery to production will happen. It's a better practice to also setup an approval process before the production deployment, so that production operators, QA people and project management can audit and control this important stage in the roll-out process. In IKAN ALM it is possible to set up approval and notification processes for every step in the life-cycle, improving the communication within the project team.

Rollback

Even if you have a well planned and executed QA and audit trail, from time-to-time things may go wrong. Therefore it is not a luxury to ensure that there is a way back, so that in the worst case scenario production service levels are minimally disturbed. IKAN ALM supports a well-defined and tested rollback process, which will protect you from nightmares like production shutdown.

Life-Cycle Management

Developing applications is not just about writing code. Once it has been developed, code must be tested, approved and delivered to the live environment. Depending on his\her profile, some users will expect their code migration to the live environment to be complete as soon as possible whereas for others it will only happen when the time is right. Therefore, a correct software change workflow process must be in place to fulfill the needs of the entire project team. IKAN ALM provides this assistance by auditing what has changed, when it changed, who changed it and who approved the change for promotion. IKAN ALM is both the glue that holds together the various components and phases of the application life-cycle and the oil that lubricates the smooth and efficient interaction of those components. It delivers an automated workflow to drive a continuous flow of activity through the development life-cycle and efficiently coordinate and streamline development changes.

Conclusion

Extending PVCS Version Manager with Continuous Integration, Life-Cycle Management and approval processes provides a flexible framework for controlling and automating the compilation and distribution of software release packages from level to level (stage to stage), site to site and across networks. As well as facilitating versioning at object and stream level, creating an automated routine of building and releasing objects. With IKAN ALM you can guarantee the quality of the code and the deployment process and provide a traceable and auditable logging for each step in the life cycle.
Final goal is a comprehensive end-to-end framework, where not only the needs of the developers are met, but also the interests of all other stake-holders.

Interfaces

You can use the IKAN ALM built-in interfaces to bridge to any of the leading versioning solutions, including IBM® Rational® ClearCase® (UCM and Base), IBM® Rational® ClearCase® LT, Microsoft® Visual SourceSafe®, Serena® PVCS Version Manager, CVS, Subversion. IKAN ALM seamlessly integrates with these versioning systems, offering you the freedom to use the IDE of your choice. It is even possible to setup mixed environments, in which one project uses sources from different versioning systems.
For more information, please visit: www.ikanalm.com

IKAN DEVELOPMENT

IKAN Development NV
Schaliënhoevedreef 20a
B-2800 Mechelen, Belgium
Tel : + 32 15 44 50 40
Fax: + 32 15 44 50 41
info@ikan.be
www.ikan.be