

IData Incorporated

Software Configuration Management Practices

A whitepaper for the 2007 Mid-Atlantic Banner Users Group

Ken Dezio and Brian Parish, IData Incorporated
10/8/2007

Software Configuration Management Practices

Definition and Background

Software configuration management (SCM) is the practice of controlling and managing software components throughout the software development lifecycle. The goal of SCM is to provide effective control over the development products that comprise the software application and to provide the ability to accurately reproduce the state of the software at any point in time. The three main areas of software configuration management are version control, change management, and release management.

- **Version control** is the management of multiple copies of a piece of software. Maintaining separate revisions of a unit of software (script, file, database object, etc) allows the software to be reverted back to a prior stable revision in the event of a problem. Version control also allows for comparison between different revisions of a unit of software to identify the changes made to the software between revisions. Versions, or revisions, of each unit of software can be managed individually by checking them in to and out of a central software repository. Tracking revisions of software units individually allows a single unit to be reverted back to a prior revision to effectively undo a change without reverting back to an earlier version of the entire software application.
- **Change management** is the practice of defining and enforcing the methodology by which changes are made to software during the software development lifecycle. Change management includes capturing information about a change, such as what change is needed, why the change is needed, where the change is to be made and the current state of the change. Within change management, an individual change should be tracked through its inception, design, development, testing, and deployment to production. Ideally, change management should be tightly coupled to version control in that the exact software units and revisions of those units changed should be recorded with the information about the change.
- **Release management** is the practice of determining which software unit revisions and changes are included in a software build and controlling when and to where the products of a software build are deployed.

SCM in the Banner Environment

Employing some form of SCM is essential when performing custom development within a complex ERP system such as SunGard Banner. This practice can range anywhere from simple version control of development products to full-lifecycle SCM.

Even if an institution does not use an automated tool to fulfill its SCM needs, it should use some combination of documented procedures and manual processes to control the development process. Maintaining separate database instances for development, test, and production along with procedures governing when changes are copied between instances will provide a rudimentary form of SCM. This practice alone may be appropriate for a small development staff managing few customizations. In this instance, rudimentary version control may be accomplished in part by periodic database backup, which would provide access to prior object versions in the event of an emergency.

The level of complexity and formality of the SCM solution used within a Banner institution should be related to the size of the development staff and extent of customization to the base product. For a single developer working on a handful of customizations, storing copies of code within a smartly-named set of directories may provide adequate control over the development products. However, as both the number of developers and the level of customization increase, the introduction of an automated SCM tool may become necessary to ensure a base level of control.

Software Configuration Management Practices

Commercial SCM Products

Several commercial off-the-shelf (COTS) SCM software packages are available and can assist in managing software changes or can be used to automate certain SCM functions. Most of the software packages are flexible and can be molded to fit an institution's existing development methodologies. The software packages range from version control systems, such as Microsoft SourceSafe, to full-lifecycle SCM systems such as Serena Dimensions.

The version control system acts as the primary interface between the software units and the SCM system. The version control system can be used alone or it may belong to an integrated package that includes change management and release control components or it may interface with third party software packages that handle those additional SCM functions. The remainder of this document will focus on version control systems and their use within custom Banner software development.

Version Control Systems

In general, version control software packages can either be server based or distributed (peer-to-peer). In server-based systems the individual software units are stored in a central software repository which resides on a designated SCM server. Developers check software units out of the repository to their local development environment, make changes, and check the modified software units back in to the repository. In a distributed system, the developer's development environment is considered a local software repository that can act as either a client or a server with respect to other local repositories within the system.

One major complication in using a version control system while developing database-based applications such as Banner is that nearly all version control software packages are file-based and work by checking software units out of the repository as files onto a directory structure within the development environment. File-based systems do not allow users to manage individual database objects unless those objects can be converted into an intermediate format and stored on the system as discrete files.

Some integrated development environments (IDEs) and other development tools have the ability to interface with third party SCM systems for streamlined version control of software units. This feature can be helpful when developing database applications as the development tool performs the task of extracting objects out of the database and checking them directly in to the version control system. One common Oracle development tool, Quest Toad for Oracle, provides the ability to interface with one of several existing third-party SCM systems. The Team Coding Facilities within Quest Toad for Oracle support interaction with third-party version control tools through CVS-based or SCCS-based interfaces. Supported third party tools include Serena PVCS, Microsoft SourceSafe, Rational ClearCase, and Starbase, among others.

Oracle provides, within its Oracle Internet Developer Suite, an SCM system, called Oracle Software Configuration Manager (SCM), which provides the ability to manage both files and database objects throughout the software development lifecycle. This is unique in that most SCM systems do not provide the ability to manage individual objects within a database. Oracle SCM is a server-based version control system with a separate Oracle database schema acting as the software repository.

Once enabled, the Oracle SCM functions can be accessed directly from within Oracle Forms Developer allowing developers to check forms data in and out of a software repository. With Oracle 9i and later versions, Oracle SCM can also be used as a stand-alone SCM system for files and Oracle database objects. Software units created outside of the Oracle Developer Suite can also be managed through Oracle SCM whether they are files residing on the file system or objects within an Oracle database.

Software Configuration Management Practices

Release Systems

When using a version control system there are three primary methods used by Banner institutions to migrate or promote custom code to a new instance (“development to test” or “test to production”). These options range from a manual process to a fully integrated system. Each process includes at least a rudimentary change management process for notification, documentation, and approval for migration of custom code.

- An example of a **manual release management** process is to control migration of code through the use of appointed gate-keepers. This means that only a small set of approved individuals has the ability to move code from one instance to another. This may also include checking code out of a version management system. These systems require that appropriate policies are created for documentation, notification, and approval of the software to be released. Frequently, the Oracle DBA group serves as the gatekeeper.
- An **automated release management** process usually relies on scripts for migrating and compiling code. Sometimes these scripts are created by the developers or DBAs, but in a truly automated system these are driven by utilities (custom or off-the-shelf) that are part of the approval and migration process. In the best of these systems, the migration works by moving files directly from the version control system.
- An example of an **integrated release management** process would be a migration that is driven by the SCM system. This means that approvals, version control, and migration are all integrated into one system.

Change Management Systems

There are two different parts of change management that will be discussed in this section. The first part is a system to keep track of information about a change. This includes ways of storing documents and “meta-data” about the project. The second part is a request and task tracking system. This includes systems for requesting changes, approving migrations, and reporting project statuses. In some instances, these two parts of change management can be contained in one integrated system.

Project and document management systems:

- **Ad Hoc systems** are used by many institutions and they most commonly are built around policies regarding storing project documents stored on shared network drives. This is a vast improvement over individuals storing copies of documents on local drives or email systems. However, it does rely on each person to remember to post the documents, and it can often have problems with multiple versions of the same document being edited at the same time. It is also very difficult to keep an updated report of all changes (summary document).
- **Web or Wiki Based Systems** are becoming increasingly popular and easy to use. These are usually rather inexpensive systems that can be installed on an institution’s own web server and allows for both project documents and statuses to be posted in a single shared space. Depending on the sophistication of the system, this can include advanced security for controlling access.
- **Collaboration Software** includes several options for off-the-shelf hosted solutions that allow for project collaboration. Some advantages of these systems include version control for documentation, regular backups, audit reports, and discussion tracking. Some example systems include BaseCamp, Central Desktop, and DocuShare (Xerox).

Software Configuration Management Practices

- Some institutions have decided to use **integrated systems** that handle documentation, task tracking, and occasionally version control as well. This can be incredibly powerful; because the systems can allow the institutions to build the change management policies directly in to the system in order to require all documentation, approvals, and version control before any project can be migrated. An example of a system used by some institutions is HP Project and Performance Management Center (formerly Kintana).

Request and task tracking systems:

- **Ad Hoc systems** used to track project requests, statuses, and migration approvals exist in many forms. One of the most common is to use request document templates that are then emailed to the appropriate gatekeepers or approver. This process makes reporting, prioritize, or collaboration very difficult. To be successful, it requires strict adherence to submission and approval policy.
- **Ticket Systems** like Remedy or Magic are used by many institutions. These Systems are excellent at keeping track of project requests and statuses. However, many of them were designed for Help Desk applications that primarily have a simple workflow of one or two steps. For software development projects there is often a need to store more request information, more complicated statuses, and workflows that include multiple steps (approvals, testing, migration, etc.).
- **A combination of systems** is often the best answer. Requests may originate with a ticket system, but then are tracked and migrating using a collaboration system, project wiki, or a fully integrated SCM.

Notes

Software Configuration Management Practices

Some Popular SCM Software Packages:

Product	Vendor	SCM Functions	Architecture
CVS	Open source	Version Control	Server-based
SCCS	Included in most flavors of Unix	Version Control	Server-based
RCS	Open source	Version Control	Server-based
Subversion (SVN)	Open source	Version Control	Server-based
Dimensions	Serena	Integrated Version Control, Change Management, and Release Management	Server-based
PVCS	Serena	Version Control with tie-in products available for Change Management and Release Management	Server-based
SourceSafe	Microsoft	Version Control	Server-based
Rational ClearCase	IBM	Version Control with tie-in products available for Change Management and Release Management	Server-based
StarTeam	Borland	Integrated Version Control, Change Management, and Release Management	Server-based
Harvest Change Manager	Computer Associates	Integrated Version Control, Change Management, and Release Management	Server-based
Bit Keeper	BitMover	Version Control	Distributed
Code Co-op	Reliable	Version Control	Distributed