Achieving developer acceptance of standardized procedures for managing applications from development to release is one of the largest hurdles facing organizations today. Establishing a standardized development-to-release workflow, often referred to as the ALM (application lifecycle management) process, is particularly critical for organizations in their efforts to meet tough IT compliance mandates. This is much easier said than done, as different development teams have created their own unique procedures that are undocumented, unclear, and nontraceable.

Keeping Score

in the IT Compliance Game

ALM can help organizations meet tough IT compliance requirements.

TRACY RAGAN
CATALYST SYSTEMS

Achieving 100 percent compliance from all development teams requires that the ALM team clearly communicate the levels of compliance to the developers and clearly communicate to upper management which development teams are and are not in compliance. Keeping track of the game using a simple "compliance scorecard" can do the job.

The first rule of thumb in defining a standardized ALM process is to clearly define the activities that fit under the ALM umbrella. For example, the ALM process does not need to define a particular development

Keeping Score in the IT Compliance Game

methodology, such as continuous development or agile development. In fact, staying away from how a development team defines the development process is highly recommended. The ALM process instead should focus on the movement of changes leaving development and flowing through test and production release. In other words, the ALM process should not be concerned about changes that a developer may make day to day. Instead, the ALM process should focus on what happens when the developer says the changes are ready to move to production.

Once the development team reports that its source-code changes are ready for prime time, the ALM workflow should focus on four specific procedures: source-code control, build management, testing, and production release. Of course, the software is initially “born” at the development stage, but the subsequent stages that involve locking down the preproduction source code, executing a preproduction build to create the binaries, testing the binaries, and finally releasing the binaries to production are the core of the ALM process that must be standardized and repeatable to meet IT compliance mandates.

Even though the core of the ALM activities involves steps after the development stage, it is critical to get the developers to accept the standardized process. This is because most development teams are accustomed to performing every stage of the ALM process from development to release. Defining a standardized ALM process has a direct consequence for the developers once they are ready to move their applications to production. It may mean, for example, that they no longer build, test, and release the application themselves, but hand it off to a separate team, or perform these activities by a particular method, using particular tools. For this reason, the development teams must accept the standardized procedures; otherwise, the ALM process can be completely derailed.

Any organization attempting to set up a standardized ALM process must identify a central team or individual with responsibility for establishing the process and implementing the new rules of engagement. In most organizations, this responsibility fits most logically into the configuration management, testing, or production

control teams. Smaller organizations may assign a single individual to assist each development team with the production release process.

KEEPING SCORE

Rolling out a centralized ALM solution and achieving 100 percent compliance from all development teams requires two critical components: the levels of compliance must be clearly communicated, and upper management must be serious about achieving the goal. This clear direction from upper management and a clear statement of the ALM process allow the development teams to understand what is expected of them. It is critical that all teams fully understand the levels of ALM compliance. It is equally critical that upper management be informed weekly regarding which teams are meeting these levels of compliance and which are not. You can use a “compliance scorecard” for communicating this information.

Defining levels of compliance is the most important

Compliance in a Nutshell

Companies of all sizes are wrestling with the issue of IT compliance. From government mandates such as the Sarbanes-Oxley Act of 2002 to meeting quality guidelines such as COBIT (control objectives for information and related technology) and ITIL (IT infrastructure library), organizations are learning to adapt their software development process so that it becomes a true business process that can be tracked, measured, repeated, and cost controlled. Even small private organizations that are not driven by government mandates are striving to reach IT compliance standards as they relate to cutting IT costs by eliminating ad hoc and nonrepeatable activities, improving software quality through metrics and measurements, and reining in the risk associated with releasing software applications to a production environment.

The trend toward IT compliance is really a way for upper management to say that software development must be managed in the same way as other departments, standardized, and repeatable. IT compliance really means that delivering business software solutions is no longer seen as a mystical activity performed by a few really technical people, but instead is viewed as a business process that must be carefully monitored, audited, and controlled to maximize the overall benefits of business automation.

step in creating your compliance scorecard. The best approach is to start with what is important to the developers. Developers will interact with the centralized ALM process in two particular areas: SCM (source-code configuration management) and build configuration management. You should select SCM and build configuration management tools that are flexible and can cater to the needs of multiple development teams. This allows each team to use the tools uniquely, depending on their needs, and still interact with a centralized process.

Developers use SCM and build configuration management tools in a closely integrated method. They need to be able to lock down source code and compile their binaries. When defining a central process, you may need to take away their team-based SCM tool; therefore, you will be taking away their build process as well. You will need to define a central process for both build and source-code management; otherwise, each team will continue to use its inferior tools. Development teams often choose SCM and build configuration management tools based on open source ad hoc scripting languages that perform the most simplistic of check-in, check-out, and build features. These tools may not perform the detailed dependency

management, impact analysis, or build audit reporting needed to trace matching source to production executables. For this reason, these tools in particular must be standardized across the enterprise ALM process.

DEFINING LEVELS OF COMPLIANCE

There is a balance between keeping developers happy all of the time and meeting the basic compliance needs of the organization. To maintain this balance, you must keep in mind the most basic needs of the development teams and make sure those needs are met in the enterprise ALM solution. In addition, you must keep the levels of compliance simple and clear. Overly complicated levels of compliance will give the developers too many excuses for not achieving them. Table 1 lists sample levels of compliance. The early levels of compliance address only SCM, and the later levels address SCM, build configuration management, and release management.

Most organizations can use the three levels of compliance shown in Table 1. These levels provide the basic framework for a strong ALM process that will meet tough IT governance standards. Each level is worth further exploration.

TABLE 1

Sample Compliance Levels

Compliance Level	Compliance Description	Compliance Benefit
Level 1	All third-party objects, SOA objects, J2EE objects, and database objects are identified and checked into the central SCM tool.	All third-party objects are centralized for the entire organization. Developers begin using the central SCM tool to manage reusable third-party objects.
Level 2	Level 1 is achieved, all project-level source code is checked into the central SCM tool, and a build of the binaries using the central build configuration management tool is executed. A traceable footprint from the build is required, validating matching source to executables.	Source code must be validated to be worth managing. Building the code with only the SCM-managed objects, with a traceable footprint based on compiler calls, will validate matching source code to production executables. This meets the separation of duties and traceability IT governance requirements.
Level 3	Levels 1 and 2 are achieved and all binaries are tested and approved for release by the central production control team without assistance from the development team. All production servers are locked down and cannot be updated by any member of the development team.	Storing the source code and building the executables must be followed up by defining a release process that does not involve the developers. This is the final way of meeting a full ALM-compliant process.

Keeping Score in the IT Compliance Game

Level 1: Identify and check in all third-party libraries to the central SCM tool. This may seem like a strange first level of compliance, but it is critical to the ultimate goal of the SCM process: providing matching source to production executables. Requiring the development teams to identify all of their third-party components, including compiler libraries, gets them involved in using the centralized SCM tool without interrupting their direct development efforts. These third-party and compiler libraries are critical for supporting the application build process, allowing for the creation of QA and production-level compiles, and validating what versions of these third-party and compiler libraries are running in production.

This level of control is critical in managing SOA (service-oriented architecture), J2EE, and database applications. Remember that many of these third-party and compiler libraries must be installed into production. The applications are built against them and execute against them in production. It is critical that the centralized SCM solution begin controlling these libraries, even before they begin controlling the application-specific code. Once the application team has defined these components, the ALM team can begin ensuring that the correct versions are running in production. In addition, as new versions of these components become available, the ALM team can control their use and distribution. The management of these third-party and compiler libraries is just as important as managing the code your developers create in-house.

Level 2: Load all application source code into the central SCM repository and recompile the application to create a preproduction level build with the central build configuration management solution. Provide the ALM team with the ability to compile the application's binary objects (.exe, .war, .jar, .dll, etc.) using the enterprise build configuration management tool on a "build" machine. The resulting footprint report shows the location of the source code that was used in the build. The last thing you want to do is to check code into the central repository that cannot compile. If the source code is not loaded into the SCM tool correctly, developers will object to using the

tool, as they will not be able to build their applications. Also, it is important that the development team establish a means to build its application in a repeatable fashion. If the team can create the build on a machine defined to do nothing but builds, it is closer to creating a repeatable build process. Additionally, your build workflow should provide the ability to create a footprint based on the actual compiler calls. This report will be crucial when completing the next step.

Level 3: Levels 1 and 2 are achieved and all releases are performed by the central production control team. Developer access to production servers is removed. This final level of compliance completes the ALM picture. Developers at this level will turn over the source code to the central ALM team, which will then create the binaries, with a traceable footprint validating matching source to executables, and will move the binaries to a secure production environment. This level of compliance is fully traceable and will meet the toughest of IT standards such as COBIT (control objectives for information and related technology), ITIL (IT infrastructure library), and SOX (Sarbanes-Oxley Act of 2002).

BUILDING A SCORECARD

Once you have met the basic needs of the developers in your ALM process and defined simple and clear levels of compliance, you will find that each team begins to debate small features of their solution compared with the enterprise solution. You must learn to differentiate between a small missing feature and a big missing feature in the enterprise-level process. In other words, the way in which a particular tool displays version history is not a showstopper. Not providing a means for developers to build their applications is a valid objection to the enterprise ALM process. If you are hearing objections from the developers regarding small features of the SCM solution, or an objection to the added work required to meet the levels of compliance, this is a sure sign that the first battle has been won. You have successfully defined a process that the developers can incorporate into their development process, leaving their team-based process behind. Your next step is to build a compliance scorecard.

The compliance scorecard is your way of communicating to upper management the success of the enterprise SCM rollout. If you have indeed completed the first step by defining an SCM process that can be used by all teams, regardless of the development tool, then it is time for the development teams to migrate to the new process.

Table 2 is an example of a compliance scorecard showing three application teams listed with three levels of

compliance. It is obvious from the scorecard that the customer service team is 100 percent compliant and that the loan processing team has not committed to a date when it will be at the highest level of compliance. This scorecard should be shared with all development team leaders.

This allows upper management to see which teams are lagging behind.

Keep in mind, not everyone can migrate at the same time. Each development team has its own deadlines, and you must work with each team to achieve these deadlines. It is not always possible to begin using a new SCM tool during a busy development cycle. It is the responsibility of team managers to give the development team the time it needs in its project schedule to participate in the migration to a new centralized process. It is not unusual for upper management, however, to put pressure on a team to get a new release out, while at the same time asking the team to migrate to a new tool that will certainly take time away from its development activities. The scorecard gives the application project managers an opportunity to negotiate when they will become compliant at the different levels, preventing any development interruptions.

There are other benefits of using a scorecard. For example, if the customer service and loan processing teams use the exact same development tools, and the customer service team is 100 percent compliant, then there should be little reason, other than project scheduling, that the loan processing team cannot use the enterprise-level process. The scorecard becomes a tool of peer pressure. Application managers should be rewarded for achieving 100 percent compliance.

SIMPLE TOOL WITH POWERFUL RESULTS

The scorecard is a simple tool that can bring powerful results. It can be created with a Microsoft Excel spreadsheet or even a database where reports can be easily obtained and viewed by upper management. As upper management begins to use this scorecard, individual managers may ask for reports showing different levels of information—for example, a report that shows all of the compliant teams, or one that shows all teams that have met the first level of compliance. The ability to customize these reports will make it easier for you to provide upper

TABLE 2

Sample Compliance Scorecard

Development Team	Level 1	Level 2	Level 3
Accts Payable	Compliant	1/24/06	6/1/06
Loan Processing	3/1/06	5/1/06	No commitment
Customer Service	Compliant	Compliant	Compliant

management with data needed for decision making.

To create some buzz around the ALM process, you must show levels of success. Providing levels of compliance that are clearly communicated and obtainable by the development teams is a successful method of getting the teams to take the baby steps needed to move from their team-based systems to the enterprise-based system. Making the first level of compliance simple will allow teams to accept the process without completely interrupting work flow. The more teams you have accepting the enterprise-level ALM process, the quicker other teams will join in. No team manager wants his or her team to be the only one that is noncompliant.

Using this scorecard method of tracking compliance can help you keep the momentum of your ALM rollout going. In addition, the compliance scorecard will keep the burden of enforcing the process with upper management where it belongs.

Finally, be realistic. The process of rolling out an enterprise ALM solution is never really done. As long as there are new development efforts, there are new teams creating their own team-based ALM processes that you will need to confront, address, and move toward compliance. ☺

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

TRACY RAGAN has served as CEO to Catalyst Systems Corporation since it was founded in 1995 and has been an active member of the Eclipse Organization since 2002. She specializes in configuration management, build management, IDEs, and deployment tools. Her position with Catalyst has provided her exposure to large financial, insurance, and embedded systems companies working on gaining Sarbanes-Oxley compliance using various version management and deployment tools in conjunction with Catalyst's flagship product, Openmake.

© 2006 ACM 1542-7730/06/0900 \$5.00