An introduction to the benefits of Application Lifecycle Management

Kobee increases team productivity, improves application quality, lowers the costs and speeds up the time-to-market of the entire application development process

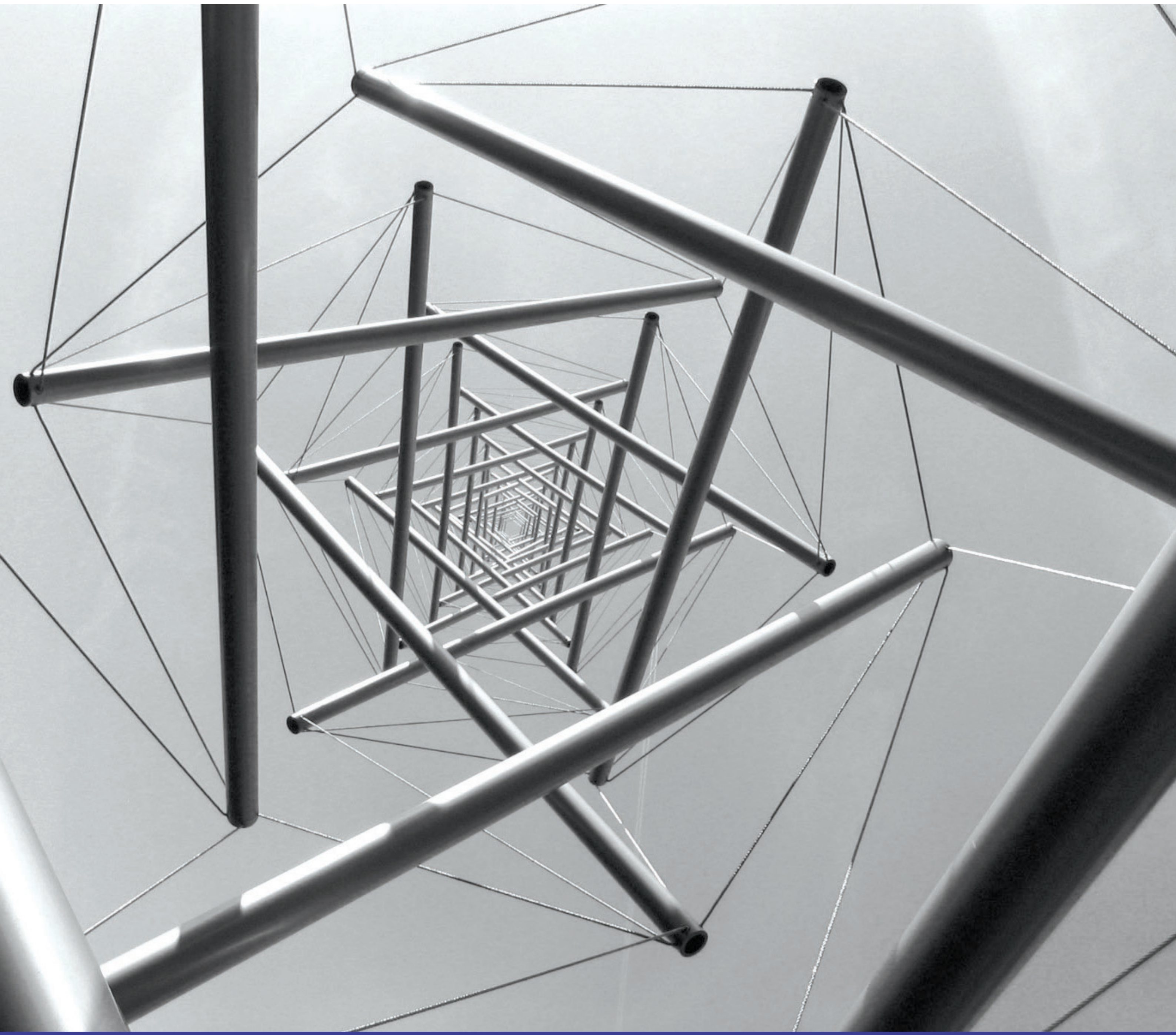


Table of contents

Why should you use Application Lifecycle Management?

Application Lifecycle Management (ALM) offers the capability to integrate, coordinate and manage all the different phases of your application development process, including the definition of requirements, design, development, versioning, build and testing stages, and, finally, the approval-based deployment to test and production.

Throughout the ALM process, each step is closely monitored and controlled using integrated approval management, issue tracking and version control.

Due to this overall approach, ALM has an enormous impact on the efficiency and costs of your application development process.

Moving forward with Kobee

Kobee is a cross-platform web-based solution for Application Lifecycle Management. It combines DevOps initiatives (continuous build and continuous integration) and lifecycle management to support the complexity of service-oriented architectures and highly distributed systems.

Kobee encompasses all aspects of the Application Lifecycle Management process offering build management, continuous integration, rebuild, manual build, release management, approval processes and deployment management. Within Kobee, you define, implement and enforce software lifecycle processes, appropriate for your organization, including development, test, quality assurance and final production.

Kobee is **methodology-independent**, **tool-independent** and **repository neutral** (enabling cross-platform development).

Kobee complies with industry standards, such as CMMI, ITIL and PRINCE. It helps your organization to enforce, control, audit, report and facilitate best practices and guidelines for Application Lifecycle Management, and provides benefits which guarantee conformity with audit requirements and improve communications with the different parties across the IT system

Higher productivity

Lower costs

Faster time-to-market

Improved Application Quality

Multi-vendor approach

In the fast-changing world of software development, the demands for smarter and more valuable tools are always on the rise. A single vendor simply cannot offer the many tools needed to cover each phase of the lifecycle. In practice, most of the software development companies use a variety of (best-of-breed) tools produced by different vendors.

The challenge is to get all those systems to work together and to fully integrate them within the overall application life-cycle. The main advantage is that users can continue working with their preferred tools and no extra costs are involved for implementing new systems.

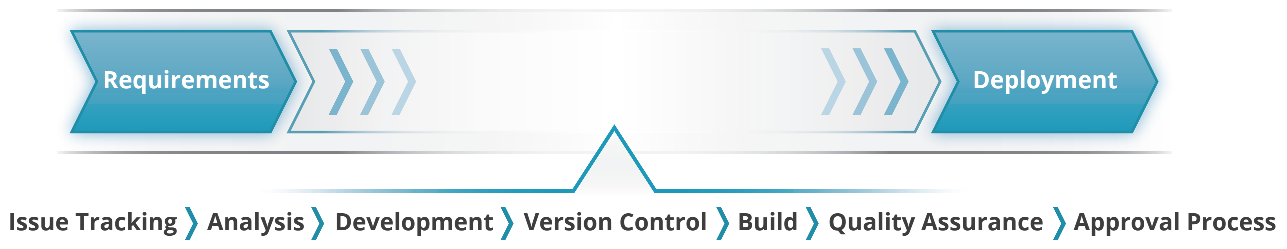
IKAN believes in the multi-vendor approach and offers a fully compliant ALM 2.0 tool. No matter what development system(s) or platform(s) you are currently using, we have an ALM framework for you.

Process

Application Lifecycle Management (ALM) is a collection of processes, roles and deliverables, which is controlled and improved with each successive iteration **i**n the software lifecycle.
Different methods can be used, such as:

- **The very strict, sequential "waterfall" method**, whereby the development process has been predefined as a succession of events from coding over testing to quality control and, finally, release of the application. In this approach, each phase of the project must have been completed before the next phase can start.
- **The spiral model**, whereby features of the waterfall method and the prototyping model are combined.
- **The incremental, iterative Agile approach**, whereby the project is split up in smaller work cycles, and the process is adapted to the evolving needs of the project.

No matter which method you use, Kobee simplifies and improves your development process.



Process phases overview

Requirements

Defining business needs and choosing matching solutions are the key issues for a successful business process. Once those issues are handled, Kobee keeps track of the implemented solutions.

Issue Tracking

Tracking and managing the issues and bugs that emerge during software engineering is a critically important task. An issue tracking system assists in creating, updating and resolving reported issues. It often also has a knowledge base containing information on customers, solutions to common problems and other data. It is a valuable asset for quality assurance and for programmers to keep track of reported software bugs and issues in their work.

Analysis

For any software engineering project – be it new application development or modification of an existing application – quality starts with analyzing the business to ensure that system requirements clearly and accurately reflect business and customer needs. Poor analysis can lead to a wide array of quality problems, including fragility, lack of scalability, and resistance to modification.

Development

This is the actual coding of a software application. Software developers may choose from a wide variety of software tools to help them build the applications, such as the modern Eclipse platform, the Microsoft's Visual Studio®.NET environment or the older C++ and Cobol development tools.

Version Control

Version control is an important part of making team-based software engineering work efficiently. Version control practices help people work on the same components in parallel, without interfering with each other's work.

Build

An important part of any software development process is the creation of reliable builds of the software. A fully automated and reproducible build, including testing, that runs many times a day, allows each developer to integrate his work on a daily basis, thereby reducing integration problems. There are several subcategories within the build process, including continuous build, rebuild, nightly build, forced build.

Quality Assurance

Too often software engineering uses quality standards that are far below those of other engineering disciplines. However, higher quality software can differentiate one company from others in its sector.

Approval Process

Reviews and approvals validate the completeness of a product and maintain consistency among its components. Controls such as electronic approval, document repositories and change-package based code reviews help to ensure that the necessary reviews can be done.

Deployment

At some point, a software application must be distributed to one or more servers at one or more locations. If the application is to meet or exceed service level agreements, then the IT operations team must assume responsibility for ensuring the quality of the roll-out. In this case deployment tasks and regular activities may be scheduled to start at any time, making it easier to coordinate deployments in advance of deadlines. This boosts development quality and productivity.

Improving the process

Each phase in the Application Lifecycle Management process typically attains a specific maturity level of process as shown on the chart.

Kobee helps your organization to achieve a higher maturity level.

Improving quality (CMMI)

The active discipline at each phase of the lifecycle defines and emphasizes quality differently because different project teams need different approaches.

Working in a team raises the need for communication and changes the way teams would execute these approaches individually. By implementing controllable, improvable and unequivocal processes to improve the total quality, the team can achieve a higher maturity level within the overall CMMI (Capability Maturity Model Integration) framework. In the long run, it is improved quality that enables software engineering teams to deliver more projects on time, at lower cost, and with more features.

By implementing the Kobee solution around best-of-breed software engineering tools, your organization's processes will evolve from chaotic and ad hoc (CMMI level 1) to continuing improvement/improving processes (CMMI level 5).

For more detailed information, visit the Software Engineering Institute's website.

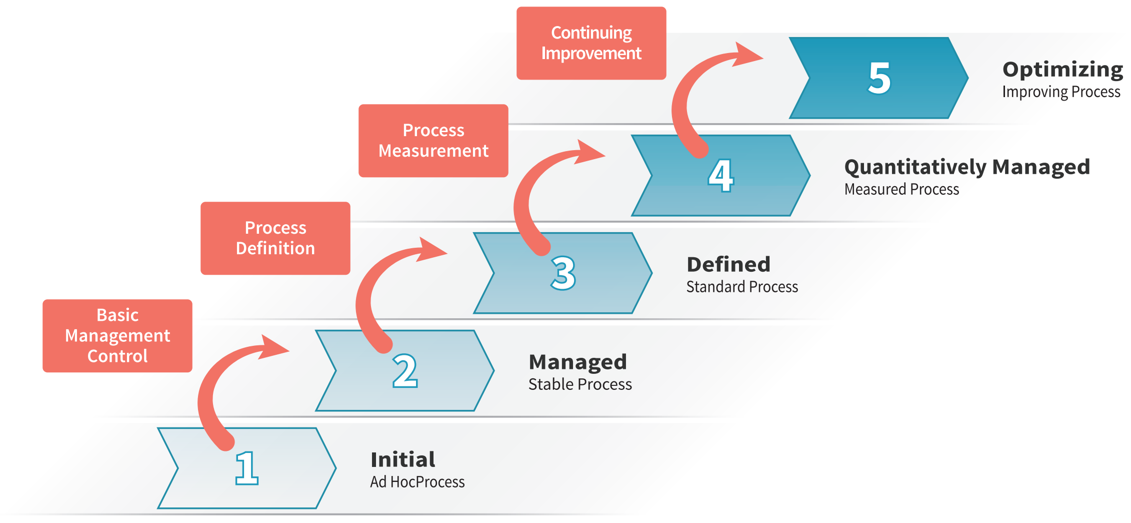
Improving traceability

It is common to have processes and controls in place for versioning.

The classic versioning systems like CVS, IBM® Rational® Clearcase (UCM and Base), IBM® Rational® Clearcase LT, Microsoft® Visual SourceSafe, Subversion, Serena® Version Manager etc. offer extensive possibilities in this field. But when building, deploying and tracking the lifecycle "pipeline" of your software applications, you lose sight of your code as versioning tools do not cover these areas. Kobee solves these issues by extending automation, control and visibility throughout the whole process, from development, through build and into production.

Overal benefits

- Fewer errors
- Faster results
- Less recursive work in QA and testing
- Predictable and shorter roll-outs
- More reliable assets in the overall IT infrastructure
- Improvement in Total Quality Management.

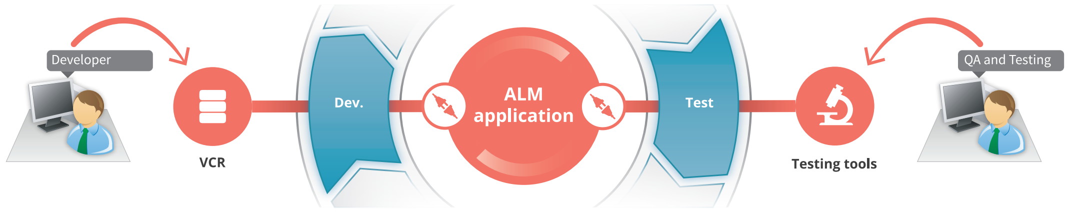


Tools

Today there are many tools that can be used in software application development. For each discipline, be it requirements management, issue tracking, integrated development environments (IDEs), versioning or testing, there is a wealth of choice.

We believe that a sound ALM solution should leave each stakeholder the freedom of choice on what tool he uses, and the ALM solution itself should handle the communication between the different tools. Kobee gives the user that freedom.

For example:

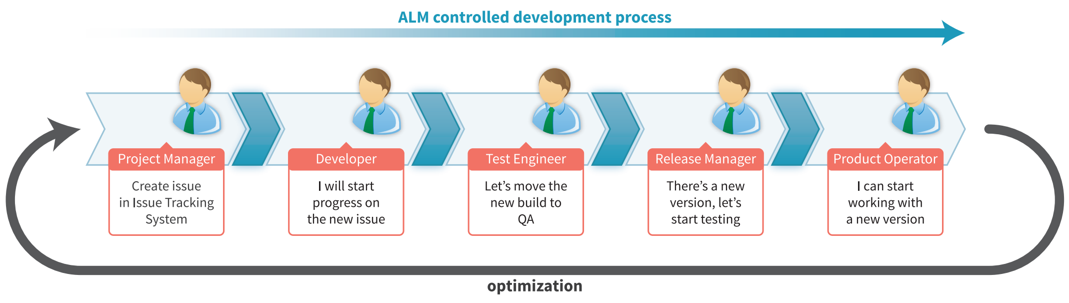


- A developer may use the IDE of his choice, as long as he can commit his code to a versioning system.
- Development teams can use any methodology, e.g., waterfall, spiral, prototyping, Agile,...
- Different development teams within an organization are able to use their versioning system of choice.
- Kobee can communicate with existing issue tracking (or defect tracking) systems.
- Local, Decentralized or Cloud development environments are fully supported.
- Kobee is multilingual and adaptable on user level.

Stakeholders

The activities of the application lifecycle involve several stakeholders, each of them focusing on different aspects of the overall development process. Implementing Kobee to manage the application lifecycle, provides a series of benefits for each group of stakeholders without compromising the responsibility of the other groups.

For example:


ALM controlled development process
Project Manager
Create issue in Issue Tracking System
Developer
I will start progress on the new issue
Test Engineer
Let's move the new build to QA
Release Manager
There's a new version, let's start testing
Product Operator
I can start working with a new version
optimization

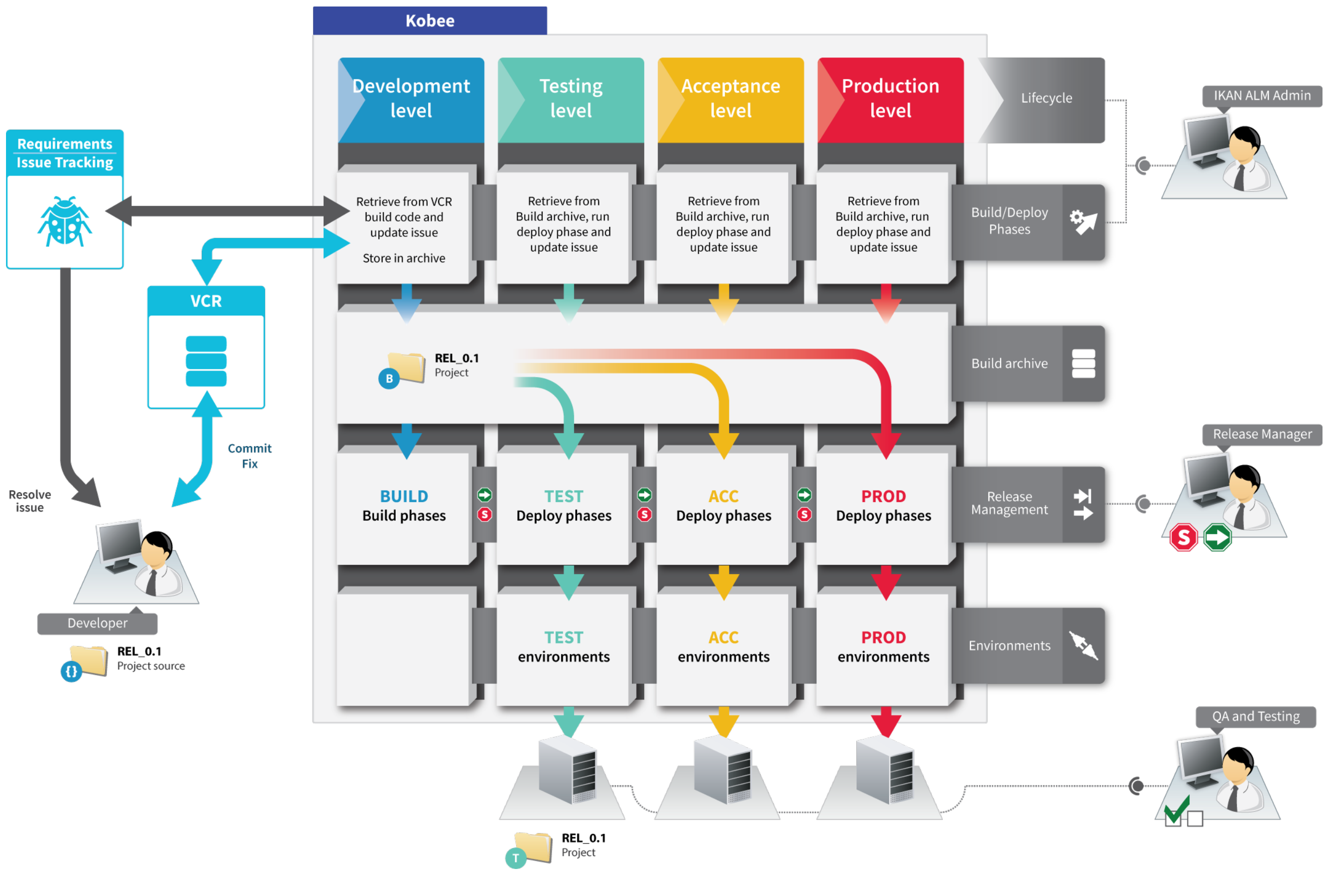
- A developer wants to have early feedback on the code committed in the trunk stream (Continuous Integration) and wants to build the project in his IDE with the correct latest sources and common libraries.
- A project manager wants to have a clear overview of the project status: is the latest code in the trunk buildable? do the unit tests run successfully? what are the guys of QA testing? which is the current production version?
- Production operators prefer an automated deploy process wherein they can control the environment variables.
- The CIO and CEO of a corporation would like to see an automated and repeatable process with an audit trail.

How Kobee works

For the following demonstration of how Kobee works, we assume that everything has already been set up (the different possibilities of Kobee and development methodologies are beyond the scope of this document). The only thing a developer needs to do is commit his code to the Version Control Repository.

Once committed, Kobee will:

- Notice the changes in the VCR
- Retrieve the code from the VCR and build the application
- Tag the code in the VCR and update the Issue Tracking System
- Deploy the application to a predefined level (e.g., Test level)
- Notify all involved stakeholders
- Wait for user approval to deploy the application to the next level (e.g., Production level)



Benefits

Every step in the ALM process solves different problems for different stakeholders. Here are some common answers.

Why use version control?

Stakeholder	Benefit
Developer	Keep track of the changes.
IT Management	Safe storage of all historic data.
Production	Easy to revert to an earlier version.
Management/Audit	No loss of data.

Why implement a continuous integration process?

Stakeholder	Benefit
Developer	Concentrate on developing software. Get early feedback on committed code.
IT Management	Get early feedback on code quality. Find weak spots.
Production	Get high-quality production code.
Management/Audit	Fewer errors. Repeatable process. Faster and shorter release cycle.

Why have an automated build?

Stakeholder	Benefit
Developer	No loss of valuable time trying to build manually.
IT Management	Allow to do more builds Give rapid feedback.
Production	Everything is coordinated by a script.
Management/Audit	Prevent mistakes.

Why approval management?

Stakeholder	Benefit
Developer	Improve communication across the project team.
IT Management	Control the evolution in the different stages of the lifecycle. Build in audit moments.
Production	Control deployment to the production servers.
Management/Audit	Traceability. Who authorized?

Why have an automated deploy?

Stakeholder	Benefit
Developer	Guarantee that production will receive the quality code that he created.
IT Management	Speed up the process and help reduce errors.
Production	No manual intervention reduces risk.
Management/Audit	Increase the release cycle frequency and productivity.

Why implement a rollback process?

Stakeholder	Benefit
Developer	More time to fix defects.
IT Management	Can always revert to the latest good release.
Production	Reduce the risk of service outage.
Management/Audit	Ensure return to exact prior state. Quickly resolve errors in production.

Why lifecycle management?

Stakeholder	Benefit
Developer	Easily build code for the test or production level.
IT Management	Have a clear view of the development process and status.
Production	Automate production deployment. Reduce the amount of rework needed.
Management/Audit	Easily answer the questions: Who? When? Why and What occurred?

Benefits per stakeholder

Stakeholder	Benefit
Development	No more repetitive or unwanted costs. Cutting costs by reusing components. No changes in working environment, no need to learn new skills. 15% more productivity.
Testing	Strong communication between Development and Testing.
Build and Release management	Complete separation of duties. No more risky deploy mistakes. 25 % more productivity in the build and release phases.
IT Management	Increased productivity. Controllable and enforceable rules for test and approval. Cost-cutting. Overall reporting on IT level. Complete project insight.
CFO (Chief Financial Officer)	Full compliance of the whole project. Included traceability of actions. Overall reporting on business level. Control over IT ownership costs.
CEO (Chief Executive Officer)	High CMMI level and control over budget and performance. Full compliance to regulators. Stronger and more flexible organization.
External (shareholders, govern-ment, auditors)	Complete and traceable compliance.

IKAN Development
Kardinaal Mercierplein 2
2800 Mechelen, Belgium
Tel. +32 15 797306

info@kobee.io
www.kobee.io