

Moving to Structured Software Change Management

In today's complex IT world, choosing the right change management system is crucial.

by Ray Bernardi
 Published 09/12/2007

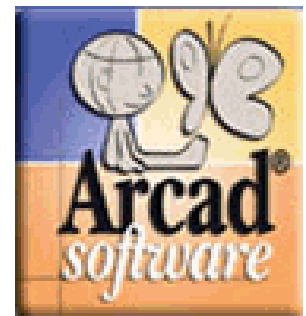
As your business needs evolve, a greater volume of software changes—possibly over several platforms and certainly under a tight schedule—are required to support your business objectives. Structured software change management, and the productivity tools it brings to the table, becomes a requirement not only for the IT staff, but for other departments within your company as well.

Over the course of my career, I've had the opportunity to see change management evolve from a manual process to one that is fully automated and traceable. I have learned that change is actually a simple process that's not so easy to do. Sure, the steps of a software change are easy enough to outline, but the actual performance of those steps is time-consuming and prone to human error.

Supervisors, managers, developers, and even an organization's leaders can benefit when a structured change management system is put in place. Most people fear change, or at the very least, resist it. You can imagine then the resistance encountered when you talk about changing the way your developers make software changes. It strikes at the very core of an organization. It examines the existing culture and points out its flaws and shortcomings.

This process of examination often uncovers many different pieces of software and many different manual operations. These all need to be integrated in some fashion in order to yield a clear picture of what's going on. Different databases from different platforms, spreadsheets, documentation, and sometimes email communications all need to be captured and brought together to know exactly what has transpired or what is now required. This integration is often a manual effort.

If you've ever spent the day constructing a spreadsheet that contains all the information you need for a meeting that afternoon, you know what kind of effort can be involved. Imagine never having to produce that spreadsheet again. A structured change management system can provide you with all the information you need at the touch of a button.



ARCAD Software, Inc.

One Phoenix Mill Lane
 Suite 203
 Peterborough, NH
 03458

Web:
www.arcadsoftware.com

Email:
sales-us@arcadsoftware.com

Tel: 800-676-4709
Fax: 603-924-7377

The reason more shops have not implemented change management varies, but the usual objections are money and time. It will cost too much, and it will take too long to implement. Those are the very things a structured change management system is designed to alleviate: the time it takes and how much it costs to make a software change. If done right, implementing change management will complement, not replace, your traditional way of doing things and will ultimately make your developers more productive. A good change management system will never overload your managers and project leaders with information or bureaucracy.

Effective change management means that changes to the system are well thought out from the beginning. Every task should be identified and evaluated prior to any code changes. There should be plans in place to recover from the change in the event things do not go as planned. A change does not end when it reaches production. A full evaluation of the change should be done to ensure it's what was requested and is in fact performing as designed.

Tackling Application Modernization

With application modernization efforts, software change management has become a lot more complex. IT organizations today are confronted with a set of new challenges. They need to deal with multiple code lines on multiple platforms and with a new skill set for developing those applications.

Companies that want to keep their competitive edge and ensure the longevity of their existing business systems need the ability to rapidly adapt their main applications to new technologies. In addition, they need to improve their applications to stay ahead of the curve.

Most companies already have software systems in place that contain their business rules. These legacy applications are vital to the company's business activities, and they're a huge investment. Applying new technologies to these existing legacy applications is the driving force behind change in a lot of organizations. This is where the right tools and a good plan come into play.

Whatever strategy you implement, you'll be confronted with four major concerns:

- Gaining a clear and accurate view of existing applications
- Maintaining stability of your applications as changes are made
- Conserving your team's productivity
- Keeping track of the systems' ever-growing complexity

How do you do you address all of these? By controlling the stages of the process. You need to define a common methodology for managing development that includes these elements:

- Auditing existing applications
- Documenting business rules
- Migrating RPG code to its more recent versions
- Merging development environments like native and Java

This is where structured change management comes into play. Poorly planned changes seldom succeed, and the decisions you make today will be around for years to come. Let's take a look at what it takes to implement structured change management over today's complex applications.

Application Analysis

The first step is to gain a clear understanding of what you have on your system now. You'll need to examine your existing environments for errors and inconsistencies that will hamper the process of making a change later on. You should be concerned with things like source code not found, source without an object, objects that are no longer used, inconsistencies on objects (such as ownership), and so on.

In addition, you may need to undertake this process on several machines. Try to identify any inconsistencies between development and production machines that could cause issues later on.

While it's typically not a requirement, cleaning up your existing environments before you implement a change management strategy is always a good idea. It is best to start the process from as clean a slate as possible.

This process need not be time-consuming. A good change management software package will have tools available to help you not only identify these situations, but also correct them.

Once you have your environments in order, it's a good idea to keep them that way. You need to define and implement a methodology that each change will follow—no exceptions. Look for a solution that will allow you to automate the entire change process across all your application servers.

From Initial Request Through Development

There are many things to consider when defining how you make changes; let's start at the beginning.

Changes start with a request for development or modification, or as an incident report or trouble ticket. Meeting regulatory demands (SOX, Basel II, etc.) further requires companies to provide total traceability for the entire chain of events, starting at this point. This means getting the users involved.

Users may need to be linked via several interfaces:

- A 5250 interface, for users with no access to PC- or Web-based solutions
- A rich-client interface installed as an executable on all PC workstations
- A browser-based interface for those who require access without a client

Users should be able to do more than simply enter and manage their own requests. They should also have access to a knowledge base for FAQs and white papers for more information when needed. All activity performed by users should be captured in the database. Look for an issue-tracking system that can produce and send emails to keep people informed along the way.

Also consider your service-level agreements (SLAs) with your user community. Any solution considered should be able to monitor a request and take action if required. This is a process known as "escalation."

Every request that enters the system needs to be analyzed and evaluated before a decision is made to

proceed. If action is required, the request then becomes a maintenance report that should ultimately be linked with the components involved, the developers who made the change, the location the changes were deployed to, and so on.

A maintenance report needs to be managed as well. A project leader should review all maintenance reports and decide how to proceed. The manager will assign these tasks to the appropriate developer or team and will need to keep track of it as it progresses through the system.

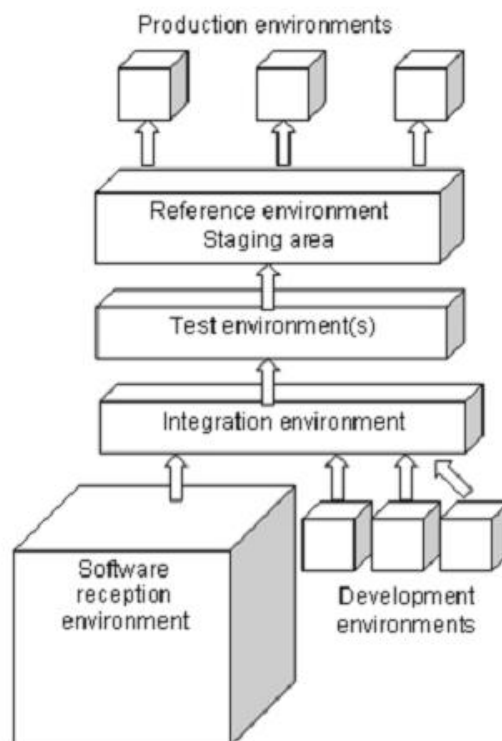
Developers working on the task need to have easy access to the maintenance report to update it with new information and to track the time involved. Of course, a capable change management system, with integrated help desk and change management modules, will provide a developer with a lot more than that.

A Repeatable Process Means Repeatable Quality

A good change management system will allow you to define many different work environments. These work environments help you structure the lifecycle of a change and provide different validation steps along the way. A solid repeatable process is required to ensure application reliability, especially during periods of intense change.

Below is a sample configuration plan:

Sample Configuration Plan



This configuration allows for the reception of code changes from outside vendors, as well as changes being made in developer environments. It includes an integration environment that serves multiple purposes. It is not only a quality assurance area, but also an area where the transfer to production process can be tested. This goes back to repeatability. If you can prove the process on a test area, then you stand a much better chance of getting it right the first time in production.

Of course, there may be more test levels at your organization. Make sure any product you consider has the capability to be configured the way you do things now. In most cases, the way you make changes now is sound. You simply need to automate it and make it auditable.

Distribution of changes locally and to remote servers should also be a function of your change management system. This distribution may be performed at different levels. For example, you may need to send changes to a testing machine and a few testing servers before you actually make the changes on your production machines. In addition, different people should be assigned to implement at different levels. A good separation of duties and responsibilities is part of this process. No one person should have the ability to alter production. It's dangerous.

Final Words of Wisdom

Make sure any software change management system you consider can handle the complex environments we see in today's IT world. As applications move onto new platforms, you don't want to keep adding tools to the process. It confuses the issue and only makes things harder. Look for a system that can work regardless of the platform.

A tool that can provide a developer with cross-platform transparency is more crucial than ever. Cross-platform capability helps in many ways. First, a developer using a tool like this can perform a complete impact analysis, thus improving effectiveness in making the change. Secondly, cross-platform capability allows perfect synchronization of different development teams and the deployment of the components they create. Make sure any solution you choose can handle new objects like ILE, SQL, Java, C++, and so on.

A good change management system will also interface well with tools that developers need, like IBM Rational.

Rational products, which incidentally were first created for the aerospace industry, provide tools for developers to use throughout the development process. Developers working in Rational environments, or Visual Studio, need the same access to the change management system as a native developer requires. Look for tools that allow you to set a methodology and support it no matter what the IDE.

Of course, change management can be done without the purchase of a tool or even the coding of a single program. It could be an entirely manual process, but when you think about the enormous number of steps involved in even a simple change, the benefits of automation become clear.



MC|iAPPLICATION DESIGNER

You'll improve your traceability, eliminate most human errors, and offload many of the tedious and repetitive tasks that your highly paid developers perform, thereby freeing them up to do what they do best: make your software better. You'll save time and money because changes can be implemented faster and recovered automatically when things do go wrong.

Having all your change data in one place, regardless of what platform the change was made on, will also ensure you can produce the reports you need, when needed, not only for management, but for those pesky auditors as well.

With a background in System i programming, **Ray Bernardi**, Solutions Architect, [ARCAD Software](#), has over 15 years of experience helping IT departments to implement the best software management infrastructure for their computing environment and to manage the transition of RPG applications to the Web.