**Welcome Guest |**

**Software Change Management**

**Change is inevitable in all stages of a software project. Change management will help you direct and coordinate those changes so they can enhance-not hinder-your software.**

Change is inevitable in all stages of a software project. Change management will help you direct and coordinate those changes so they can enhance—not hinder—your software.

The only constant in software development is change. From the original concept through phases of completion to maintenance updates, a software product is constantly changing. These changes determine whether the software meets its requirements and the project completes on time and within budget. One of your main goals as project manager is to manage software change.

**Change Management and Configuration Management**

Your project probably has software configuration management (SCM) in place. If designed well, SCM is a major component of software change management. All too often, however, SCM is an add-on process, focused primarily on capturing the software’s significant versions for future reference. In the worst cases, SCM functions as a specialized backup procedure. If SCM is left at this low level, the unfortunate project manager can only watch the changes as they happen, preach against making bad changes, and hope the software evolves into what it should be. Of course, evolution is difficult to predict and schedule.

Software change management is the process of selecting which changes to encourage, which to allow, and which to prevent, according to project criteria such as schedule and cost. The process identifies the changes’ origin, defines critical project decision points, and establishes project roles and responsibilities. The necessary process components and their relationships are shown in Figure 1. You need to define a change management process and policy within your company’s business structure and your team’s development process. Change management is not an isolated process. The project team must be clear on what, when, how, and why to carry it out.

|  |
| --- |
| **Figure 1:** Software Change Management |
| http://twimgs.com/ddj/sdmagazine/images/sdm9907e/9907ef1.gif |

The relationship between change tracking and SCM is at the heart of change management. SCM standards commonly define change control as a subordinated task after configuration identification. This has led some developers to see SCM as a way to prevent changes rather than facilitate them. By emphasizing the change tracking and SCM relationship, change management focuses on selecting and making the correct changes as efficiently as possible. In this context, SCM addresses versions, workspaces, builds, and releases.

A change data repository supports any change management process. When tracking changes, developers, testers, and possibly users enter data on new change items and maintain their status. SCM draws on the change data to document the versions and releases, also stored in a repository, and updates the data store to link changes to their implementation.

Software change management is an integral part of project management. The only way for developers to accomplish their project goals is to change their software. Change management directs and coordinates these changes.

**Where Changes Originate**

A variety of issues drive software changes. Understanding the origins of prospective changes is the first step in prioritizing them. The sources of change can be classified as planned development, unexpected problems, or enhancements.

***Planned Software Development.*** Ideally, all software change would result from your required and planned development effort, driven by requirements and specifications, and documented in your design. However, adding new code is a change you must manage. Adding functions that were not requested (no matter how useful and clever) consumes project resources and increases the risk of errors downstream. Even requested features may range in priority from “mandatory” to “nice to have.” Monitoring the cost to implement each request identifies features that adversely affect the project’s cost-to-benefit ratio.

***Unexpected Problems.*** You will undoubtedly discover problems during any development effort and spend resources to resolve them. The effort expended and the effort’s timing need to be proportional to the problem—small bugs should not consume your project budget.

The team must determine whether the code fails to implement the design properly or whether the design or requirements are flawed. In the latter case, you should be sure to correct design or requirements errors. Integrated change management toolsets, which I’ll discuss later in the article, can make the process seamless: change to a code file can prompt the developer to update the corresponding documentation files. The investment in documentation updates will be recovered many times over when the software is maintained later.

***Enhancements.*** All software projects are a research and development effort to some extent, so you will receive enhancement ideas. Here is where project management is most significant: the idea could be a brilliant shortcut to the project goal, or a wrong turn that threatens project success. As with requirements or design errors, you need to document these types of changes. Adhere to your development standards when implementing an enhancement to assure future maintainability.

**Critical Decision Points in Change Progress**

You should address changes when they are only potential changes, before they’ve consumed project resources. Like any project task, changes follow a life cycle, or change process, that you must track. In fact, three critical decision points, as shown in Figure 2, drive any change process. These decision points form the framework of change management.

|  |
| --- |
| **Figure 2:** Change Decision Points |
| http://twimgs.com/ddj/sdmagazine/images/sdm9907e/9907ef2.gif |

***Approve the Concept.*** Change requests come from testers or users identifying problems, and from customers adding or changing requirements. You want to approve all changes before investing significant resources. This is the first key decision point in any change management process. If you accept an idea, assign a priority to ensure appropriate resources and urgency are applied.

***Approve to Proceed.*** Once you’ve accepted a change request, evaluate it against your project’s current requirements, specifications, and designs, as well as how it will affect the project’s schedule and budget. This analysis may convince you to revise your priorities. Sometimes, the team will discover that a complex problem has an elegant solution or that several bugs have a common resolution. The analysis will also clarify the cost-to-benefit ratio, making the idea more or less desirable. Once you clarify the facts, make sure the change is properly managed with a second formal review.

***Approve the Resolution.*** A change request is completed when the change is folded into the planned development effort. During requirements analysis and design phases, this may occur immediately after you approve the request. During coding, however, you often must conduct separate implementation and testing to verify the resolution for any unplanned changes, including both testing of the original issue and a logically planned regression test to determine if the change created new problems.

After testing, you must still review the change to ensure it won’t negatively affect other parts of the application. For example, the developer may have changed a correct user prompt to match the incorrect software logic. If the testing indicates a risk of further problems, you might want to reject the change request even at this point.

***Rejected or Postponed Requests.***At any of the decision points, you can decide whether to reject or postpone the change request. In this case, retain the change request and all associated documentation. This is important because if the idea comes up again, you need to know why you decided against it before. And, if circumstances change, you may want to move ahead with the change with as little rework as possible.

***Emergency Processing.***If a problem has shut down testing—or worse, a production system—you may not have time for a full analysis and formal decision. The change management process should include an emergency path distinct from the flow shown in Figure 2, with shortened analysis and streamlined approval. Focus this process on an immediate resolution, whether a code “hack” or a work-around, that eliminates the shutdown. You can update the change request to document the quick fix and change it to a lower priority. By leaving the change request open, you won’t omit the full analysis and resolution, but you can properly schedule and manage these activities. Alternately, you can close the emergency change request when the fix is in place, and create a new change request to drive a complete resolution.

**Roles and Responsibilities**

The change management process requires several decision-makers at the various decision points. Your change management process should address the following questions:

• Who will make the decision? Ultimately, the project manager is responsible for these decisions, but you can delegate some of them to other project leaders.

• Who must give input for the decision? Who can give input?

• Who will perform the analysis, implementation, and testing? This can be specified generally, although each issue may require particular contributors.

• Who must be notified once the decision is made? When, how, and in how much detail will the notice be given?

• Who will administer and enforce the procedures? Often this becomes a task for SCM or the release manager, since it directly impacts their efforts.

You don’t need to handle all issues at all project stages the same way. Think of the project as consisting of concentric worlds starting with the development team, expanding to the test team, the quality team, and finally the customer or user. As your team makes requirements, design, and software available to wider circles, you need to include these circles in change decisions. For example, accepting a change to a code module will require retesting the module. You must notify the test team, who should at least have a say in the scheduling. The standard SCM baselines represent an agreement between the customer and the project team about the product: initially the requirements, then the design, and finally the product itself. The customer must approve any change to the agreed-upon items. The change management process helps you maintain good faith with the customer and good communication between project members.

**Change Management Tools**

Because of the volume of data involved, you often need tool support to manage software change. As with any type of tool, you should get the right tool for your job. Your process should drive the tool; don’t expect the tool to solve the problems alone. Unfortunately, you often don’t know what process you want until you’ve tried using the wrong tool. Keep in mind that if you’re producing software now, you have at least one process already at work. Identifying the best current process and the problems with it are the first steps to defining a better process.

A successful system coordinates people, process, and technology. Once you define the process and tools, ensure that your team is trained and motivated to use them. The best tool is worthless if it is not used properly, whether from lack of skill or resentment over being forced to use it. Process and tool training should make the tool’s benefits clear to your team.

Change management’s most important components are an SCM tool and a problem-report and change-request tracking tool. Increasingly, change management toolsets integrate with one another and with development tools such as requirements or test case tracing. For example, you can link a new version directly to the change request it implements and to tests completed against it.

At the simple and inexpensive end of the tool scale are SCCS (part of most UNIX systems) and RCS, which define the basics of version control. Various systems build on these, including CVS and Sun’s TeamWare, adding functions such as workspace management, graphical user interface, and (nearly) automatic merging. In the midrange are products such as Microsoft’s SourceSafe, Merant’s PVCS, MKS Source Integrity, and Continuus/CM, which generally provide features to organize artifacts into sets and projects. Complete SCM environments are represented by Platinum’s CCC/Harvest and Rational’s ClearCase, giving full triggering and integration capabilities.

**SCM Tools**

SCM tools range from simple version engines, like SCCS, to sophisticated environments, like Rational’s ClearCase, that provide for all SCM functions. Generally, the most significant selection factor is the complexity of your development plan: how much parallel work the tool must support and how many versions it must track. If your project involves changing the same code in two different ways simultaneously (for example, maintaining the production version while developing the next release), carefully review how the tool handles branches and merges. Most tools lock files while they are being updated; if simultaneous change is your norm, look for tools that provide either a change-and-merge or a change-set process model. Performance and scalability are also issues for large projects. The larger the number of files in your project, the more you need features like directory archival and logical links between artifacts. These links that let code updates prompt the developer to update documentation. With a large project team, you need triggers to automate notification and other coordinated actions.

You should go into demos with a sketch of how your development process works, especially if you’re considering a significant tool expenditure. This lets you ask specifically how the tool could handle your needs. The tool budget will need to include the effort to define and document procedures, write scripts and integration artifacts, and train the team. If the tool is new to your organization, verify that the vendor can support your implementation or recommend a consultant who can.

**Problem-Report and Change-Request Tracking**

The key to a good issue tracking system is the ability to tailor it to your process and standards. Every project tends to want different report fields, called by different names, taking different values. Too much variation from these expectations cause even a good tracking tool to seem counterintuitive and frustrating. If your team doesn’t like to use the tool, you won’t get the complete tracking that you need. If you currently have a tracking system (even paper-based), use it as a pattern for what you want. If you’re starting from scratch, think through the change process and ask what information the participants need.

As with other tools, estimate the volume of data the tool needs to handle and verify that it will perform at that level. Consider how many individuals need to use the tool at one time and whether you need strict controls over who can change various parts of the data. If you conduct your reviews in meetings, report generation will be a significant part of tool use. For an electronic approval cycle, the e-mail interface is vital. Increasingly, tools are providing a web interface to simplify distributed use.

**Key to Change Management**

Change management lets you control software evolution and provides the basis for metrics and process improvement. Data collected under a consistent process supports estimating and planning, reducing risk, and making development more predicable. In the long run, managed change reduces the time to market, improves quality, and increases customer satisfaction. By understanding the origins of change, the critical decision points, and the roles in the decision process, you will gain enough control to manage, rather than just watch, software change.

|  |  |
| --- | --- |
| **CHANGE TRACKING TOOLS AND SCM TOOLS** | |
| **Continuus/CM**(SCM) **Continuus/PT** (Change Tracking) Continuus Software  Irvine, Calif.  Tel: (949) 453-2200  [www.continuus.com](http://www.continuus.com/) | **SourceSafe**(SCM) Microsoft Corp. Redmond, Wash. Tel: (800) 426-9400 [msdn.microsoft.com/ssafe](http://msdn.microsoft.com/ssafe) |
| **Source Integrity Professional**(SCM and Change Tracking) MKS Inc.  W. Waterloo, Ont. Canada  Tel: (800) 265-2797 [inquiry@mks.com](mailto:inquiry@mks.com) | **PVCS** (SCM and Change Tracking) Merant (Intersolv/Micro Focus) Beaverton, Ore.  Tel: (503) 645-1150  [pvcsinfo@intersolv.com](mailto:pvcsinfo@intersolv.com) |
| **RCS**(SCM) Free Software Foundation Boston, Mass.  [gnu@gnu.org](mailto:gnu@gnu.org) | **ClearCase**(SCM) **ClearDDTS**(Change Tracking) Rational Software Corp.  Lexington, Mass.  Tel: (617) 676-2400  [info@rational.com](mailto:info@rational.com) |
| **CCC/Harvest** (SCM) **Apriori**(Change Tracking) Platinum Technologies Oakbrook Terrace, Ill.  Tel: (800) 442-6861  [www.platinum.com](http://www.platinum.com/) |  |