# Managing Change in Software Development Projects with a Virtual CCB

**Ivo M. Michalick Vasconcelos**
**Project Director, Vetta Technologies**

## Abstract

Software development projects (SDPs) tend to have an unusually high degree of ambiguity and change in their requirements. Their scope is many times a "work in progress", while change is always part of the equation that leads to success. This paper describes a workable and agile mechanism to handle change in such projects, based in the concept of a Virtual Change Control Board (VCCB).

The VCCB implementation presented in this paper is supported by an array of automated tools, such as a web-based issue tracking system, text files, spreadsheets and a configuration management system. It can serve as a reference to the implementation of a change management system in a typical software development organization.

## Introduction

"**The only thing that is constant is change itself**." (Unknown author, but likely a project manager)

Software development projects (SDPs) tend to have an unusually high degree of ambiguity and change in their requirements. Their scope is many times a "work in progress", while change is always part of the equation that leads to success. This paper describes a workable and agile mechanism to handle change in such projects, based in the concept of a Virtual Change Control Board (VCCB).

Nowadays many SDPs are developed in different and remote locations, sometimes even offshore. They also usually span multiple time zones, making face-to-face meetings expensive and hard to schedule. This issue, coupled with a high degree of change in requirements, demands that SDPs use an agile, and mostly virtual, process to manage scope changes.

We present in this paper a change management process that is currently in use and that has been used successfully in other SDPs in the past. It is supported by an array of automated tools, such as a web-based issue tracking system, text files, spreadsheets and a configuration management system. It can serve as a reference to the implementation of a change management system in a typical software development organization.

## Characteristics of SDPs

SDPs have some specific characteristics that must be considered in their management process. They are usually complex, have a higher degree of uncertainty and their products need to be easily adaptable to different uses. All these characteristics contribute to making change management an important part of their overall management process. Let´s take a closer look on these characteristics.

### Complexity

The complexity of a large software is usually higher than that of other modern artifacts, and is directly linked to its number of components and the array of relationships among them. One single line of source code (SLOC) can be translated to hundreds of lines of machine code in the target machine where the software must run. In order to implement one single requirement, hundreds or even thousands SLOCs may be needed.

Complex software can have tens of millions SLOCs. The Windows NT 3.1 operating system, released in 1993, is said to have 6 million SLOCs, and the Windows Vista Beta 2 version, released in 2005, 50 million SLOCs (Wikipedia, *Source lines of code*).



Originally published as a part of 2007 PMI Global Congress Proceedings – Cancun, Mexico

### Uncertainty

Robert Wysocki (Wysocki, 2006, p. 14) considers complexity and uncertainty positively correlated with one another. Hence, complex SDPs also present a high degree of uncertainty. Also, there are literally countless ways of implementing a set of requirements for a SDP, depending on factors of choice such as:

- Programming language;
- Operating system;
- Target hardware;
- Software framework;
- Among others.

### Adaptability

Any software that gets to be actually used by its users is subject to adaptation throughout its life cycle. Such changes, if not handled properly, can inject errors in an operating software, and that can be hard to fix. Also, as the business environment that generated the need for the software changes during its development, changes may need to be made to the software requirements while it is still being build. In order to avoid scope creep such changes need to be properly managed and a scope baseline must be kept current.

## Change Management in SDPs Performed for External Clients

Many organizations need to use SDPs in order to achieve a set of strategic goals, but at the same time do not have software development as one of their core competencies. In such cases it is common practice for these organizations to keep a team of software and procurement specialists that contract such projects from different vendors. That means acquiring software to be developed under contract by an outside stakeholder, what demands a contract management system and a very well detailed Statement of Work (SOW), "a narrative description of products, services, or results to be supplied" (PMI, 2004, p. 376).

As part of a contract management system, a change management system is needed in order to keep track of all change requests handled during the project life cycle. Many of these changes may need to be paid for by the customer, so this process demands a certain degree of formalism and record keeping, based mostly on written communication.

## Change Management according to the PMBOK Guide

The PMBOK Guide (PMI, 2004) covers change management extensively, and in a more focused way in its "**Integrated Change Control**" process, in the Monitoring and Controlling process group. It states that a change management process needs to include activities such as (pages 96-97):

- "Identifying that a change needs to occur or has occurred.
- Influencing the factors that circumvent integrated change control so that only approved changes are implemented.
- Reviewing and approving requested changes.
- Managing all approved changes when and as they occur, by regulating the flow of requested changes.
- Maintaining the integrity of baselines by releasing only approved changes for incorporation into project products or services, and maintaining their related configuration and planning documentation.
- Controlling and updating the scope, cost, budget, schedule and quality requirements based upon approved changes, by coordinating changes across the entire project.
- Documenting the complete impact of requested changes."

The PMBOK Guide also proposes that project change requests be handled by some authority, usually called **Change Control Board (CCB)**. The roles and responsibilities of a project CCB need to be defined during project planning and agreed upon by all relevant stakeholders, such as customer representatives, project sponsor and project manager. In order to properly implement a change management process and a CCB some tools are usually required.



Originally published as a part of 2007 PMI Global Congress Proceedings – Cancun, Mexico

Exhibit 1, extracted from PMI´s "Standard for Project Configuration Management" (PMI 2007, p. 17), illustrates a process proposed by PMI in order to execute configuration change management.

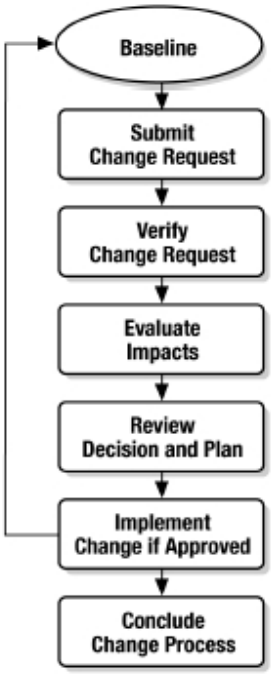


Exhibit 1 – Change Control Process according to PMI´s "Standard for Project Configuration Management"

The process proposed in this paper and described in the next session was based upon the process depicted in Exhibit 1.

## Process Description

Our process starts with a **project baseline**. It is composed of items such as:

- **Project management plan**, that includes a description of the change management process and the composition and attributions of the project CCB;
- **Project requirements,** usually captured as Use Cases and their respective descriptions and derived from the initial Project Statement of Work (SOW);
- **Project schedule**;
- **Project size estimates,** usually made using **FPA** (**Function Point Analysis**);
- **Project budget/contract**. It is common, in projects developed for external customers, to make provision in the contract for how change requests shall be priced. That usually includes a price for **function points** (a unit of project size/complexity) and another for **effort hour** to be considered when evaluating change requests.

Most projects need also to have a **traceability matrix** of some sort, mapping requirements to use cases, functional units, test cases or other artifacts relevant to software development. That is an important aid in evaluating the impact of change requests.

### Step 0.Initial setup

At the end of project planning, an initial project baseline needs to be created using an automated tool for configuration management. All work products generated during the project life cycle are stored and controlled using this tool.



Originally published as a part of 2007 PMI Global Congress Proceedings – Cancun, Mexico

Also, the change management process defined for the project needs to be mapped and configured as a project in a web-based change management tool. That allows remote use by team members and authorized customer representatives, regardless of their physical location. Such tool is the main resource to implementing a **virtual version of a CCB** (a **VCCB**).

Another important communication tool is a mailing list software, that keeps record of every email message sent to the project email address and forwards it to every subscriber to the project list – these subscribers need to be identified and listed in the Communication section of the Project management Plan.

The change management system can be configured to handle different kinds of changes, such as change requests, bugs (that need to be fixed) and issues. We will focus our description in how such system handles **change requests** (also called **CRs**). One important aspect regarding CRs is defining who has the authority to propose one. In projects developed under contract these subjects are usually the **project manager for the performing organization** (we will call her **P-PM**), and a **project manager from the customer organization** that acts as its representative throughout the whole project (we will call her **C-PM**). Both of them are part of the CCB and need to get its approval for all proposed changes.

### Step 1.Submit Change Request

All change requests must be submitted by the C-PM or by some other customer representative designated in the Project Management Plan. All change requests must be recorded in the change management system and receive a unique ID provided by it. They need to have a clear text description and, optionally, can have file attachments, such as a proposed change for a previously approved requirement or use case.

The change management system is configured in such a way that the P-PM receives an email notification every time a CR is submitted or changed.

### Step 2.Verify Change Request

The P-PM, upon receiving email notification that a change request has been proposed, checks the request in the change management system. If needed, he contacts the C-PM or the change requestor to get additional information, and decides whether the change request process shall proceed, since sometimes change requests are made in error or are considered to be invalid.

### Step 3.Evaluate Impacts

The P-PM, after accepting a new change request, assigns it to be evaluated by a team member. This is usually (but not always) the team member responsible for the module or use case for which the change has been requested. This team member needs to prepare an impact evaluation, in terms of effort hours needed to implement the change (including tests) or change complexity in function points (that usually is the case for more complex change requests). This impact evaluation is recorded in the corresponding change request record in the change management system. Sometimes the team member needs extra help in evaluating change requests, and such help needs to be requested from P-PM. That needs to be planned ahead, and a provision of about 2% to 5% of the project team´s working time is allocated to the change management process. Depending on the project, prior approval of the P-PM may be required **before** the impact evaluation is entered in the system.

### Step 4.Review Decision and Plan

The C-PM must approve the impact evaluation for the process to proceed. If she does so (by registering her opinion in the corresponding change request record in the change management system), the process continues. At this point what usually happens is that, periodically or upon request, the P-PM prepares a plan for all change requests already evaluated and approved. This plan is discussed in the CCB (not necessarily via a presential meeting, it can be done via conf call, for instance, with the proposed plan being sent previously via email) and, if it is approved, the process moves to the next step.



Originally published as a part of 2007 PMI Global Congress Proceedings – Cancun, Mexico

It is important to remember that, when a group of change requests is approved, there will usually be some impact to the project schedule and budget/contract (the customer will likely pay for the changes). So, the approval of both the C-PM and the P-PM is mandatory in this step.

### Step 5.Implement Change if Approved

The agreed upon set of change requests is implemented, tested and incorporated in the project baseline. After successful delivery of each change request, its corresponding record in the change management system is moved to a CLOSED/VERIFIED state.

## Supporting Tools

The whole process described in the previous section is supported by an array of open software tools, such as:

- Change management system: **Bugzilla** (Bugzilla);
- Mailing list software: **Mailman** (Mailman);
- Configuration management system: **CVS** (CVS).

It is also recommended that the project uses a web-based tool to publish and exchange project artifacts, such as status reports, meeting minutes and project deliverables. That can be implemented by using a **Wiki, "**a web application designed to allow multiple authors to add, remove, and edit content" (Wikipedia, *Wiki*).

## Final Words

We described a process that is in use for managing change in software projects developed for external customers. It implements a change management process that is adherent to PMI recommendations (PMI 2004 (PMI 2007), and is based on a set of open software tools. Such process is mostly virtual, so it can be implemented in projects that use virtual teams or that are developed remotely, even offshore. This can result in lower costs on trips and, more integration between teams remotely scattered and a better control of changes in project scope.

This process, although supported by a set of automated tools, relies heavily on people interactions, especially between the project managers on both sides of the project (supplier and customer). A good and noise-free communication channel between these two subjects has been identified as an important critical success factor.

The process described in this paper can be used as a starting point for implementing a change management process in a typical software development organization**.** The process relies heavily on automated tools, and the list of tools presented here can be taken as an initial recommendation.

## References


*Bugzilla*. Retrieved 30/6/07 from the Bugzilla website: **http://www.bugzilla.org/** .

*CVS*. Retrieved 30/6/07 from the CVS website: **http://www.nongnu.org/cvs/** .

*Mailman*. Retrieved 30/6/07 from the Mailman website: **http://www.gnu.org/software/mailman/** .

PMI – Project Management Institute (2004). *A Guide to the Project management Body of Knowledge Third Edition*. Newtown Square, Pennsylvania: PMI.

PMI – Project Management Institute (2007). *Practice Standard for Project Configuration Management*. Newtown Square, Pennsylvania: PMI.

*Source lines of code*. Retrieved 30/6/07 from the Wikipedia website: **http://en.wikipedia.org/wiki/Source_lines_of_code**

*Wiki*. Retrieved 30/6/07 from the Wikipedia website: **http://en.wikipedia.org/wiki/Wiki**

Wysocki, Robert K (2006). *Effective Software Project Management*. Indianapolis, IN: Wiley.




Originally published as a part of 2007 PMI Global Congress Proceedings – Cancun, Mexico