Chapter XI

Change and Closeout Management

You must welcome change as the rule, but not as your ruler.

(Denis Waitley)

Change is a fact of life for most projects, particularly IT projects. *The biggest single cause of project overruns is changes in scope* (Hallows, 1998). Change can be for the good or the bad, but change is to be expected, and change has to be managed. This chapter is concerned with that management process.

Project Changes

Resistance to change is usually rooted in fear of the unknown as PMs and people in general prefer stability and predictability. However, in order for a project to survive, PMs must deal with changes effectively. Many projects fail not in their goals and directions, but in their plan (or lack of a plan) for dealing with changes. Not having a formal change control system "guarantees a project will be plagued by chaos, errors, permanent damage, low productivity, and unmanageable software evolution" (Brown, 1998). *Formal change control is vital in projects involving an external performing or benefiting organization so that one organization can be appropriately compensated for the additional effort.*

Change can arise for many reasons; in IT projects, the leading sources or indicators of change are:

- The customer (benefiting organization) is unsure of their needs
- The customer is unsure as to how the needs should be delivered
- The performing organization is unclear to the details of the customer's needs
- The performing organization is not sure how to do the work
- The deployment environment has changed
- Planned methods or algorithms prove unsuitable
- Better methods of building or deploying the system have come forward
- The business case for the project has changed
- Market demographic and/or geographic shifts
- The sociopolitical environment for the project has changed
- The corporate environment has changed (reorganizations, mergers, acquisitions)

Changes in a project can come in any part of the project from early planning through project closeout; in IT projects, most changes come later in a project such as during implementation, testing and deployment. *Changes coming later in a project are usually much more expensive than if that need for a change were identified earlier.* A change in one part of the project deliverable may cause changes in many other parts of the project as well. So, as was discussed earlier in this book, it is very important to take steps early to flush out user requirements and potential changes. Validation of the preliminary product manifestations with quality stage gates using methods as prototypes, use case walkthroughs with customers, and design reviews with appropriate stakeholders will minimize changes later in a project.

Project change management usually concerns changes to scope, but other changes also need to be managed. Normally, change control systems are set up to deal with only scope and deliverable change; other project change is usually handled via risk management, as was discussed earlier in this book. Scope management in general includes the processes necessary to make sure that all project work is addressed and that extra work is not done. Scope change control should be planned and procedures defined early in the project (charter, SOW, contract, overall project plan). Change control is concerned with the following (PMI, 2000):

- Influencing the factors that cause change control to ensure that changes are beneficial
- Determining that a scope change has occurred
- Managing the actual changes when and if they occur

In the IT environment, the term *change control* can be confusing because those words are often applied to the control of changes to program source code and/or changes to a target hardware/software deployment platform. In this book, the control of source code and related artifacts will be called version control, and the control of hardware and dependent software components on the deployment platforms will be called configura-

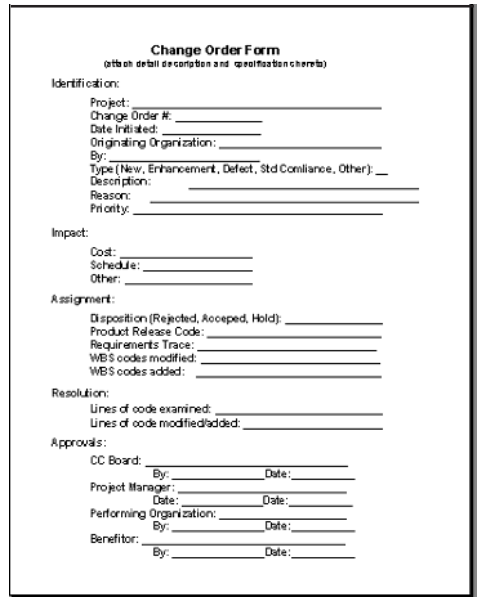
tion control; both of these topics will be discussed later in this chapter. In this book, the term *change control* refers only to change control at the project level; however, a change in a project requirement or deliverable may trigger corresponding version or configuration changes.

Establishing a Change Control System

A project scope change control system defines the procedures by which the project scope may be changed and typically includes forms, tracking systems, and approval levels. An illustration of such a form is shown in Figure 11.1. It should be integrated with a system that logs and documents all project changes. A code or number should be assigned to each proposed change whether the change is proposed internally (i.e., project team member) or externally (i.e., customer). This degree of formality is usually imposed after a segment of the methodology is initially completed. For example, formal change control of the design work is normally implemented after the initial design is completed. The pros and cons of the change should be analyzed and discussed in terms of all project parameters including cost and time consequences, and what other changes may be necessary (or desirable) due to this change. The propose change may or may not be within the project budget, or may or may not be part of the undistributed budget.

The discussion of suggested changes normally takes place at a regularly scheduled meeting of a change control board, which has members from both the performing

Figure 11.1. Change order form



Change Order Form
(attach detail description and specification sheets)

Identification:
Project: _____
Change Order #: _____
Date Initiated: _____
Originating Organization: _____
By: _____
Type (New, Enhancement, Defect, Std Comliance, Other): __
Description: _____
Reason: _____
Priority: _____

Impact:
Cost: _____
Schedule: _____
Other: _____

Assignment:
Disposition (Rejected, Accepted, Hold): _____
Product Release Code: _____
Requirements Trace: _____
WBS codes modified: _____
WBS codes added: _____

Resolution:
Lines of code examined: _____
Lines of code modified/added: _____

Approvals:
CC Board: _____
By: _____ Date: _____
Project Manager: _____
Date: _____ Date: _____
Performing Organization: _____
By: _____ Date: _____
Benefitor: _____
By: _____ Date: _____

organization and the benefiting organization. It is also advisable to include representatives from other major stakeholder groups. Representatives should be educated into the process that they are about to become a part of. To maximize the benefit of these meetings and minimize the meeting time lost by attendees, the meetings should not be too frequent, briefing packages should be sent to representatives in advance of the meeting, and changes should be sorted and bundled into product functional areas. Often, many small changes can be analyzed and discussed as one change set or release. Defects and noncompliance to requirements and/or standards normally do not go through formal change control, however in some organizations the change control board may be advised of such issues during the project. *Sometimes there is a fine line between a change to the scope and noncompliance to the original requirements; and of course this may cause conflict between the organizations involved.* It is best to document all such issues and their resolution even if that information will not be turned over to the change control board.

If the change is approved by the board, then there may be certain required additional levels of approval necessary in the line management of the performing and benefiting organization. Often if a change is within the undistributed budget, then higher approval is not required. After approval, a new work breakdown structure (WBS) work packet is created (or an existing packet modified) to reflect the change and a new project baseline cost plan and schedule is created; for externally contracted work, a contract change may also be required, which is called a "change order." Figure 11.2 shows an example of change control information (change control number) associated with WBS work packets. Usually it is better to create a new WBS work packet that will be a replacement (or in addition to) another packet; for replacements, the old packet has its planned cost set to zero (or to the amount already spent on it if work on it was already started).

Figure 11.3 illustrates a typical overall procedure for handling project changes. After a change request is initiated, a decision is made as to whether the requested change is actually part of the original (baseline) scope or not. This decision may be made by the CC board or upstream from the board by project management. Those change requests that are encompassed by the baseline scope are sent to work control to be incorporated

Figure 11.2. Form to add WBS Code (showing change info)

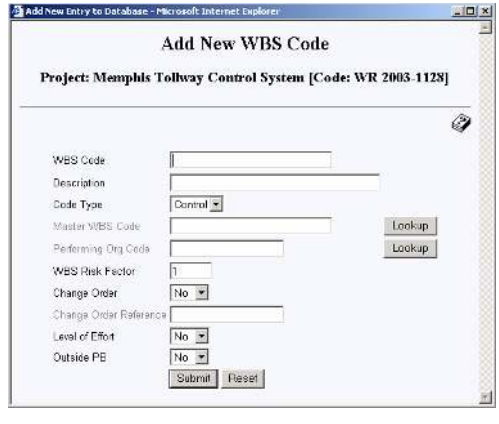
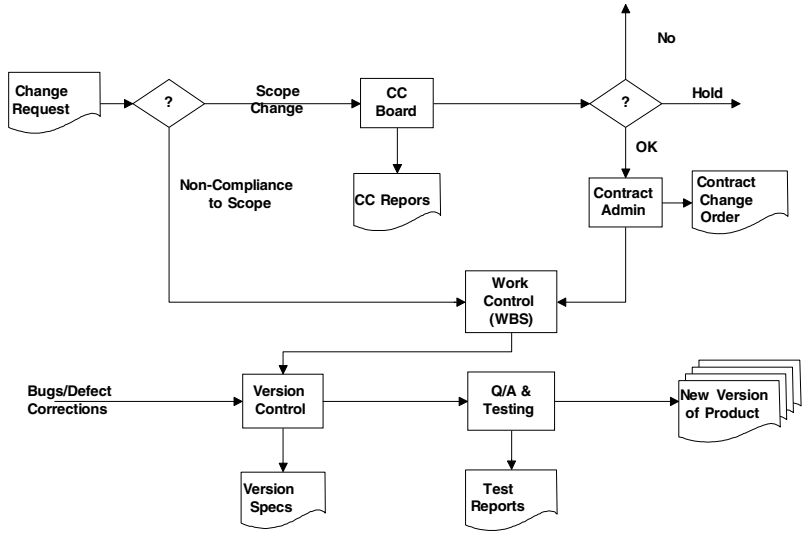


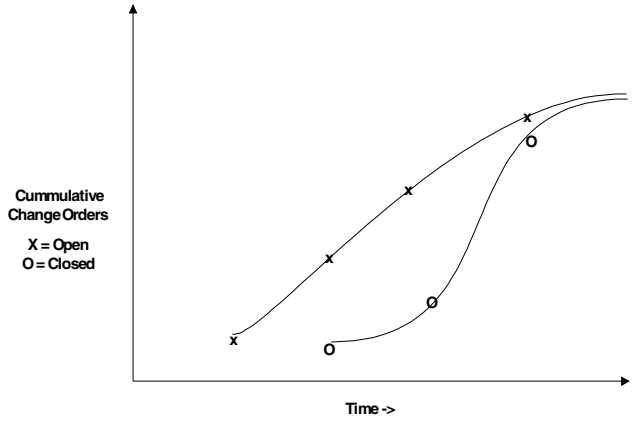
Figure 11.3. Change control process



into the WBS. Those change requests that are for additional work are sent to the CC board for study. The board studies the request in light of benefit versus project impact and makes a decision to implement the change or not. If the decision is to implement, then contract administration may need to price the new work and issue a contract change order. If the contract change order is approved (by benefiting and performing organizations), then the change goes to work control. Scope changes and corrective changes are typically batched into product versions which go through version control (discussed later) and Q/A.

Figure 11.4 shows a useful type of trend chart for cumulative change orders opened and cumulative change orders closed (approved for implementation or rejected). For this type

Figure 11.4. Cummulative change orders



of analysis, one hopes to see the gap between open change orders and closed orders closing as the project proceeds. If that gap (between the two curves) is becoming wider, then there was a problem in the project scope definition. For IT projects that succeed, 70% of the cost of the product is in the maintenance phase; thus, maintainability is extremely important. A measure of IT product maintainability is the lines of code affected per change order. A comprehensive metric in this area includes both the lines of code affected and the lines of code examined per change order, because in implementing a change, programmers spend about half of their time just examining the existing code. Both of these metrics are included in the sample change order form of Figure 11.1.

Version Control

Version control is primarily a process for making sure that multiple developers do not override each others changes to source code modules. It typically involves manual or automatic check-in/check-out procedures for these modules. Version control may apply to IT project artifacts or deliverables other than just source code including various forms of documentation (including design documents) or system content. In a modern software engineering process, where requirements and design as well as implementation are captured in rigorous notations early in the life cycle, formal version control is applied to all of these artifacts (Royce, 1998). Version control systems typically document changes in terms of who, what, why, and when changes were made, as well as keeping developers from stepping on each other toes. Version control systems may also maintain information on who can make and/or approve what type of changes and maintain archive copies of all versions of artifacts, handle naming and numbering rules, thus providing traceability of the software and related artifacts throughout the product's life cycle. Version control systems typically manage changes to source code and related artifacts both due to project level changes (requirement changes and additions) and due to internal changes such as bugs or needed redesigns. Modern version control products also coordinate changes so that builds of a particular version use the appropriate and current modules. Traditionally the building process for a software product involved compilation and linking operations; however, today, with modern Web-based systems, there may not be any linking or any compilation. However, there are still processes (Java bytecode production, encryption [HTML, JavaScript, PHP], obfuscate, etc.) that need to be performed in making the transition from the source code the developers use to the code that is placed on the production server(s).

The major components of a version control management system are (Brown, 1998)

Item Identification

- Product identifiers (names and numbers)
- Product artifact identifiers
- Identification of product acceptance criteria
- Identification of changes

- Identification of releases
- Change Control
  - Change criteria
  - Revisions
  - Procedures
  - Organizational review
- Status/Accounting
  - Product description records
  - Change status records
  - Verification records
  - Authorizations and approvals
- Audits
  - Formal Qualification Reviews
  - Physical Audits
  - Functional Audits

There are number of commercial products available for version control some of which are server based and some are peer-to-peer based, these include QVCS (www.qumasoft.com), CS-RCS (www.componentsoftware.com/), Code Co-op (www.relisoft.com/), Surround SCM (www.seapine.com/surroundscm.html), and CodeMatrix (www.codematrix.com/). There are also a number of free, shareware, or open source products available (www.thefreecountry.com/programming/versioncontrol.shtml).

Configuration Control

Control of interdependent software modules and hardware components on client and server platforms is typically called configuration control. However, sometimes the term *configuration control* is applied to both *version control* (as discussed previously) and *deployment platform configurations.* For a client server application, there may be many interdependent software modules each of which requires particular versions of the other modules to operate. This may include the operating systems, database management systems, device drivers, communication protocols, and many middleware software components. This is further complicated because there may be many versions of a particular software interdependent modules in time and/or in features. For example, a software product may be in its fifth version, with eight different customizations installed on thousands of customers' computers, where each customer may have a different version and customizations.

With the advent of Internet applications, a single network protocol (TCP/IP), and thin clients, organizations were able to eliminate all of their configuration control issues on

the client side (because the only application level client software required was a browser). In addition, many configuration control issues on the server side were also eliminated since once the new version of an application was placed on the server it was downloaded to clients when they first accessed the system (downloaded HTML, JavaScript, CSS, and image or other multimedia files). However, with the need (or opportunity) to sell goods or services globally now, configuration control issues have resurfaced in a different form since the web content may have to be "localized" for different languages, countries, and demographics.

Scope Creep

Scope creep is very common in IT projects. Customers should get what they asked for and expect, no more and no less. Giving the customer extra work is called gold plating. However, because the majority of IT projects do not succeed, you should never do additional work for the customer because you may not be compensated for it, and you do know for sure that the customer actually wants the work done. Scope creep is so common because it is natural for project team members to want to please their customer, and much of the communication between project team members and customer personnel takes place without any involvement of the PM. It is also not uncommon for a project member (particularly in IT and in other technical work) to want to explore technical details beyond the task boundaries. The PM must make sure that all the stakeholders know what is encompassed in the scope and also what is *not* part of the project scope; this distinction is best made early in the project. *The project team members (and PMs) have to be educated (and reminded) about the need to prevent scope creep and for formal change control.*

The Software Program Managers Network (Brown, 1998) lists best practices for change and version control:

- Make change management everyone's job
- Create an environment and process that enables change management
- Define and document the process, then select a tool set to support it
- The change control staff should consists of individuals with technical expertise to support the development and maintenance of the product
- The change plan and procedures need to be developed and documented in the same way that the development plan is created at the start of the project

They also establish the following rules for successful change management (Brown, 1998):

- The change management system must "own" the product information
- Early identification and control of products and artifacts is essential

- The change management process and procedures must be simple and supported by the methods and tools used in the product development
- The actions of the change control board must be documented
- Change control must be a primary focus and integrated into the organizational culture
- All information that is placed under change control must be "promoted" via a quality (stage) "gate"
- All proposed changes must be properly classified (as to the type of change)
- All releases of code and artifacts from the library (check-outs and check-ins) must be recorded
- It is essential that the change control system be continually aware of the status of the products/artifacts under control and of the relationship of one product/artifact to another
- The worst way to establish change control is to buy a tool set and then to fit your process to the tool

The Software Engineering Institute's (SEI; www.sei.cmu.edu/cmm) CMM defines necessary Level 2 practices for software configuration management (change control):

- Are change control activities planned for the project?
- Has the project identified, controlled, and made available the software products through the use of configuration management?
- Does the project follow a documented procedure to control change and configuration?
- Are standard reports on the software baseline distributed to affected parties?
- Does the project follow a written policy for implementing change control activities?
- Are project personnel trained to perform those change control activities?
- Are measurements used to determine the status of change control activities?
- Are periodic audits performed to verify that software baselines conform to the documentation in which they are defined?

There are also industry standards (IEEE 828) and military standards (MIL-STD 973, 2549) that can be used as further guidelines in setting up change, version, and configuration control systems. Software systems are available to aid in the control of project change management. As well as features built into general project management software systems such as FiveAndDime, there is software dedicated to project change management such as TrackWise (www.sparta-systems.com), SeaPine (www.seapine.com/cmsuite.html), Software Planner (www.softwareplanner.com), and ChangeManagementExpert (www.change-management-expert.com/).

Project Closeout

Management must, at some point, stop taking change orders, and the project must officially end. Additional requests for changes can be deferred to a successive version of the product or deferred until the product is officially transferred to the operation and maintenance phase. The original contracting arrangement (written or implied) will often determine (or suggest) the time a project should end. This is usually after all the original scope is completed plus important change orders that must be done before the product is placed in service. In many IT projects, coming to closure may be difficult:

- Sometimes a project will continue when funding is available, even when the budget and schedule is far exceeded; this is often due to management's misunderstanding of "sunk costs" and/or a lack of a clear original business justification.
- Sometimes it is difficult to call an end to projects when the benefiting organization stills wants more changes (and is willing to pay for them) and the performing organization wants to do the work.
- Sometimes the buyer (which is often the benefiting organization) will not sign off for official acceptance until some issue(s) is resolved.
- Sometimes the project team wants to continue because some of them may be concerned about: future assignments, significant scope has not been completed yet, documentation deliverables have not been completed, and bugs or problems are still present (even though they may not yet have surfaced in testing).

Project closeout can be a very hectic time and can also be a time of distress or excitement for different stakeholders. The PM must take care that all tasks and issues are fully and properly finished. Team members may worry about future work after the project is complete or they may be overly eager to get on with the next project. But unless the projected is ended, and a new one started if necessary, the work being done will no longer match the original goals and business justification.

Projects may come to successful completion and end normally or they may be terminated abnormally. A project may end normally (perhaps within time and budget constraints) even though management may decide not to use the product (put it into operation). This decision may be for a number of reasons, including:

- The product is no longer needed or wanted by the end users.
- Economic conditions have change, and the business value can no longer be realized.
- The product cannot be operated in an economically feasible manner.
- Resources have run out.
- The product is not sufficiently secure.
- The product cannot be economically maintained.

- The product is not sufficiently "usable."
- The product has major design flaws.
- The product is riddled with bug.
- The product does not comply with current standards or regulations.
- The project sponsor ("champion") no longer supports the effort.
- Other alternative products have proven more effective.

These conditions could have also occurred at a stage gate review, and the project could have been prematurely canceled or put on hold. This criteria are part of a quality stage gate review as defined in this book, and organizations can save considerable amounts of money and effort by discovering any of these problems before the project fully completes.

Whenever a project is canceled or put on hold, the PM should consult with key stakeholders and administrative offices (e.g., legal, HR, and procurement/contracting offices) to determine the impact of the cancellation and mitigating actions that may be necessary. To minimize organizational impact on both the performing and the benefiting organizations, a formal cancellation plan may be needed, and all stakeholders should be informed of the cancellation at the appropriate time and in the appropriate manner. The cancellation plan should also include steps to salvage reusable components.

When the project is completed, either normally or abnormally (hold or cancel), some key activities that should be performed include:

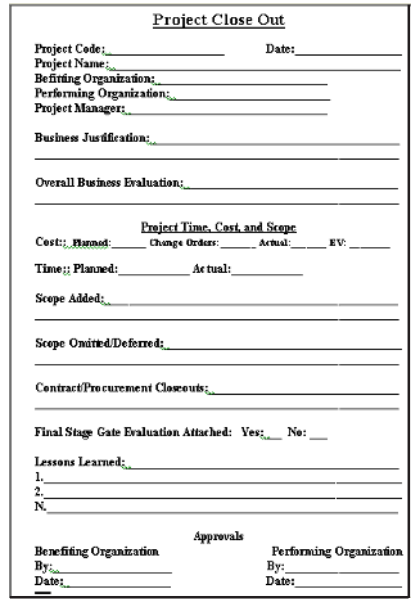
- Procurement audits and contract(s) closeout
- Product validation and verification, including change orders
- Confirmation of deliverables
- Formal inspections and compliance verification by regulators
- Formal acceptance and sign off" by buyer or benefiting organization
- Notification of completion to all stakeholders
- Formal "turnover" to operations/customer
- Postproject review and lessons learned documentation
- Itemization of change orders that were not done (deferred or rejected)
- Updating and closing project time, cost, and other records
- Properly archive of all records and artifacts
- Turnover relevant information to PM of next phase/version of product
- Acknowledgment of support of key stakeholders
- Acknowledgment of outstanding contributions
- Releasing project resources, including financial accounts and security items
- Formally releasing team

Project closeout forms or checklists are often used. Figure 11.5 is an example of a simple project closeout form. In some organizations, project closeout is divided into three parts: administrative closeout, contract closeout, and personnel closeout. Administrative closeout is done by the PM and project team in conjunction with line management of the performing organization and ensures that all project matters, including all billing, have been completed. Contract closeout is done by the procurement office and ensures that all contracts are properly completed, all vendors are paid, and all inventory issues are settled.

Personnel closeout is done by the HR office, which handles final personnel evaluations (typically done by the PM), return of checked-out equipment and other items, and final security issues. Project team members (groups or individuals) need to be properly released when their role on the project is over. If they are released in a proper and timely manner, costs will be reduced by not having to make work for them until they are reassigned to other projects. This also improves morale by reducing uncertainty about future project assignments or work opportunities. A final team meeting should be held so that all closing activities can be coordinated and a post mortem done that answers the following questions:

- Was the business justification realized (or does it appear that it will be realized)? If not, why not?
- What processes, methods, tools, techniques, and resources worked well?
- What processes, methods, tools, techniques, and resources did not work well?

Figure 11.5. Project closeout form



Project Close Out

Project Code: _____ **Date:** _____
Project Name: _____
Benifiting Organization: _____
Performing Organization: _____
Project Manager: _____

Business Justification: _____

Overall Business Evaluation: _____

Project Time, Cost, and Scope

Cost:: Planned: _____ **Change Orders:** _____ **Actual:** _____ **EV:** _____

Time:: Planned: _____ **Actual:** _____

Scope Added: _____

Scope Omitted/Deferred: _____

Contract/Procurement Closeouts: _____

Final Stage Gate Evaluation Attached: Yes: ___ **No:** ___

Lessons Learned: _____
1. _____
2. _____
N. _____

Approvals

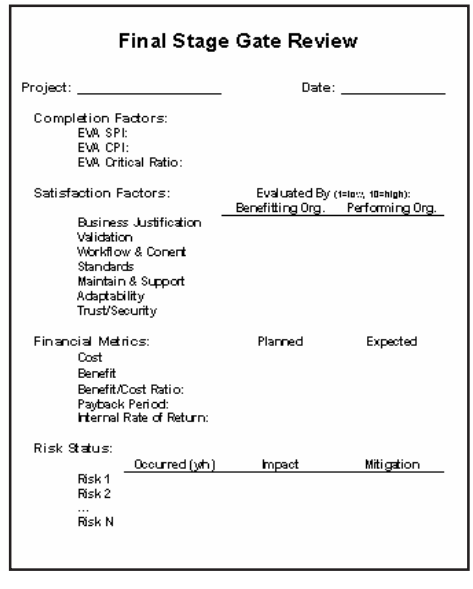
Benefiting Organization	**Performing Organization**
By: _____	**By:** _____
Date: _____	**Date:** _____

- What risks events occurred, and how they were handled?
- What risks events did not occur? Why not?
- What artifacts and components can be reused?
- What in general should be done differently on the next such project?

All these questions must be answered and documented. In addition, this documentation must be formulated in such a manner that it will actually be used. The answers to this post mortem can be entered into a lessons-learned software system. This process, as well as knowledge management in general, is discussed further in Chapter XVI.

A final stage gate review would be necessary if a stage gating process was used. An example of a stage gate review form is illustrated in Figure 11.6. This final quality stage gate review provides the final view of the project's critical success factors: completion and satisfaction. Whether or not a final stage gate report is used, it is common for a final project closeout report to be created. That final report includes the information shown in Figures 11.5 and 11.6, including final progress, time, and cost. Often a separate closeout meeting is held with the benefiting organization and possibly with the end users. In this closeout meeting, the end users are introduced to the operation and support people with whom they may work in the future. Often the performing organization uses this meeting as an opportunity to introduce or discuss the possibility for product extensions or enhancements. For successful projects, many organizations also have a celebration party for the project team and other key stakeholders. For unsuccessful projects, many organizations have an outside independent auditor review the way the project was managed.

Figure 11.6. Final stage gate review form



Final Stage Gate Review

Project: _____ Date: _____

Completion Factors:
EVA SPI:
EVA CPI:
EVA Critical Ratio:

Satisfaction Factors:	Evaluated By (1=low, 10=high): Benefitting Org.	Performing Org.
Business Justification		
Validation		
Workflow & Conent		
Standards		
Maintain & Support		
Adaptability		
Trust/Security		

Financial Metrics:	Planned	Expected
Cost		
Benefit		
Benefit/Cost Ratio:		
Payback Period:		
Internal Rate of Return:		

Risk Status:	Occurred (y/n)	Impact	Mitigation
Risk 1			
Risk 2			
...			
Risk N			

Chapter Summary

In this chapter, project general change management has been covered, particularly for IT projects version control and configuration control. A serious problem for IT projects is always "scope creep," and this topic was also covered. Project closeout and related topics were also included and illustrated.

References

Brown, N. (1998, November). *Little book of configuration management.* Software Program Managers Network.

Hallows, J. (1998). *Information systems project management*. American Management Association.

PMI. (2000). *The Project Management Body of Knowledge (PMBOK)*. Project Management Institute. ISBN 1-880410-22-2.

Royce, W. (1998). *Software project management.* Addison-Wesley.