CHAPTER 5

Defining and Managing Project Scope

CHAPTER OVERVIEW

Chapter 5 focuses on developing a scope management plan to define and manage the project and product deliverables of the project. After studying this chapter, you should understand and be able to:

- Identify the five processes that support project scope management. These processes, defined by the Project Management Body of Knowledge (PMBOK), include initiation, planning, scope definition, scope verification, and scope change control.
- Describe the difference between product scope (i.e., the features and functions that must support the IT solution) and project scope (i.e., the deliverables and activities that support IT project methodology).
- Apply several tools and techniques for defining and managing the project's scope.

1 GLOBAL TECHNOLOGY SOLUTIONS

On Friday evening Matt and Kellie were still at the GTS office, working on the project charter and project plan for Husky Air. Rubbing her eyes, Kellie asked, "I know we defined the goal of the project by developing the MOV, but what about the work that has to be done to get us there?"

"Glad you asked," Matt said as he put down his personal digital assistant on the desk in front of him. "I think we're ready to start defining the scope of the project, which will help us define all of the deliverables and activities that support the MOV."

"I remember working on a project that never seemed to end," she replied. "The users always wanted to add more bells and whistles to the system, and the project ended up missing its deadline and costing a lot more than we had planned."

Matt thought for a moment, then asked. "So, what can we learn from that experience?"

Kellie smiled. "First of all," she said. "I think we need to have a plan in place to make sure that the scope of the project is well-defined. I think part of our problem was that we never really got a clear idea of the project's goal; so we never defined the scope of the project properly. And secondly, we should've had some kind of process in place to control scope changes once we started the project."

Matt agreed. "That sounds like an excellent idea. But why not just say no to any scope change requests?"

Kellie sat back in her chair. "The way I see it, if we say yes to each and every scope change request, we run the risk of escalating the project's schedule and, in turn, the project's budget. On the other hand, if we say no to all scope change requests, we run the risk of missing some opportunities or appearing non-responsive to our client's needs."

"Good point, but how do you know when to say yes to a scope change and when to say no?" asked Matt.

"I guess we could let the project's MOV be our guide," she answered. "If a scope change supports the MOV, then it's worth doing. Otherwise, if it doesn't support the MOV, then the scope change isn't worth the time or money. Besides, the client has to make the decision whether the change in scope is worth the increase in schedule and cost. All we can do is keep the schedule and budget under control and then point out to them how any requested scope change will impact the project."

Matt stood up, saying "I think what we need is a scope management plan that can be part of the project charter. It's also important that we let everyone involved with the project know about it. Let's call it a day and get started on it first thing Monday morning."

Kellie agreed. "That's the best idea I've heard all day. Why don't you go ahead? I'll lock up after I make a few phone calls."

Things to Think About

1. What is the importance of ensuring that the scope of a project has been defined completely and accurately?
2. What is the relationship between the project's MOV and its scope?
3. What is the importance of having scope control procedures in place?
4. Why should a scope control process be communicated to all of the stake holders of a project?

INTRODUCTION

This chapter focuses on defining and managing the work that must be accomplished by the project team over the course of the project. The term **scope** is used to define the work boundaries and deliverables of the project so what needs to get done, gets done—and only what needs to get done, gets done. Therefore, it is important to define not only what is part of the project work, but also what is not part of the project work. Any work not part of the project is considered to be outside of the project's scope.

Project Scope Management Processes

The Project Management Body of Knowledge (PMBOK) defines five processes to support the knowledge area of project scope management, as shown in Table 5.1. This process group begins with a **scope initiation** process whereby the project sponsor gives

Table 5.1 Scope Management Processes

Scope Management Process	*Description*
Project scope initiation	Ensuring that authority and resources are committed to developing a scope management plan.
Scope planning	Setting the scope boundary to determine what is and what is not included in the project work.
Scope definition	Identifying the product and project deliverables that support the project's MOV.
Scope verification	Confirming that the project's scope is accurate, complete, and supports the project's MOV.
Scope change control	Ensuring that controls are in place to manage scope changes once the project's scope is set. These procedures must be communicated to all project stakeholders.

the project manager the authority and resources to define the scope of the project. In the context of the IT project methodology, the authority to commit time and resources to define the project's scope is included in the second phase when the project charter and project plan are developed.

Once the commitment and resources to develop the project charter and plan are in place, the next process focuses on scope planning. This planning process entails setting the boundary of the project in order to determine what is and what is not to be included in the project work.

The third process centers on scope definition. While scope planning defines the project boundary, scope definition identifies the project deliverables (as identified in the IT project methodology) and the product deliverables (the high-level functionality or features of the IT product to be delivered by the project team). As a result, the boundary and deliverables defined by the scope planning and definition processes provide a key component for developing the project charter and plan. Moreover, the boundary and deliverables become critical inputs for estimating the project's schedule and budget.

Once the scope is defined, the process of scope verification confirms that the scope is complete and accurate. The project team and sponsor must agree to all of the project deliverables. This not only sets expectations, but also focuses the project team on what needs to get done and what is outside the scope of the project.

Time and resources will be wasted needlessly if the scope of the project is never defined accurately or agreed upon. However, changes to the scope may be inevitable as new information becomes available or if the needs of the organization change. Therefore, a process called scope change control is needed to handle these changes so that if a scope change is appropriate, the change can be approved in order to amend the project's schedule and budget accordingly. In addition, scope change control procedures also protect the scope boundary from expanding as a result of *increasing featurism*, requests by project stakeholders to keep adding additional features and functions (i.e., bells and whistles) to the project once the scope has been set. Remember that the scope, schedule, and budget relationships suggest that increasing the project's scope (i.e., expanding the scope boundary) will generally require an increase in schedule and budget. Therefore, adding additional work to the project's scope will ultimately lead to a project that misses its deadline and costs more than originally estimated. Subsequently, once the project's

scope has been set, approved changes to the project's scope must be reflected in the project's baseline plan.

Together, the processes and techniques for defining and managing scope make up the **scope management plan.** Depending on the size and nature of the project, this plan can be separate and/or summarized in the project charter. Regardless, the procedures for defining and managing the scope of a project must be communicated and understood by all of the project's stakeholders to minimize the likelihood of misunderstandings. Moreover, the project's scope must align and support the project's MOV. Why spend time and resources to perform work that will not add any value to the organization or help the project achieve its MOV? Again, work that does not add value consumes valuable time and resources needlessly. Figure 5.1 summarizes the components and processes of a scope management plan.

PROJECT SCOPE INITIATION

Scope initiation provides a beginning process that formally authorizes the project manager and team to develop the scope management plan. In terms of the IT project methodology, this authorization is given after the project is formally accepted and funds are committed to developing the project charter and plan by the project sponsor or client. The business case provides important information about the project's description, MOV, risks, assumptions, and feasibility. In addition, the business case provides information about the background of the project in terms of why it was proposed and how it aligns with the organization's overall strategic plan.

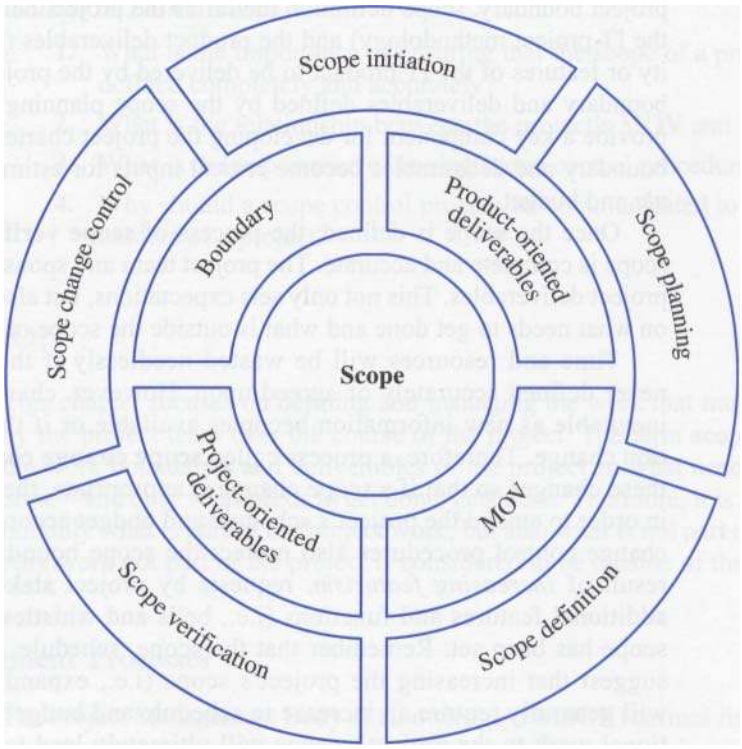

Figure 5.1 Scope Management Plan

Project **Scope Planning**

> Failure to define what is part of the project, as well as what is not, may result in work being performed that was unnecessary to create the product of the project and thus lead to both schedule and budget overruns.
>
> Olde Curmudgeon, *PM Network Magazine,* 1994.

Scope planning is a process for defining and documenting the project work. More specifically, a project's scope defines all the work, activities, and deliverables that the project team must provide in order for the project to achieve its MOV. It is an important step in developing the project plan since one must know what work must be done before an estimate can be made on how long it will take and how much it will cost.

Scope **Boundary**

Defining the scope boundary is the first step to establishing what is, and what is not, part of the project work to be completed by the project team. Think of the scope boundary as a fence designed to keep certain things in and other things out. As Figure *5.2* illustrates, any work within the scope boundary should include only the work or activities that support the project's MOV. This work is what we want to capture and keep within our fence. On the other hand, a project team can spend a great deal of time doing work and activities that will not help the project achieve its MOV. As a result, the project will consume time and resources with very little return. Therefore, the scope boundary must protect the scope from these activities once it is set and agreed upon by the project stakeholders. Having a clear and agreed upon definition of the project MOV is critical for defining and managing the scope boundary.

The Scope **Statement**

One way to define the scope boundary is to create a **scope statement** that documents the project sponsor's needs and expectations. For example, let's say we are outside consultants hired to develop an electronic commerce application for a bank. After developing and presenting a business case to our client, we have been given the authority to develop the project charter and plan. Although the business case provides a great deal of relevant information, we will still set up several meetings and interviews with key stakeholders in the bank. Based upon these meetings and interviews, we create a scope statement.

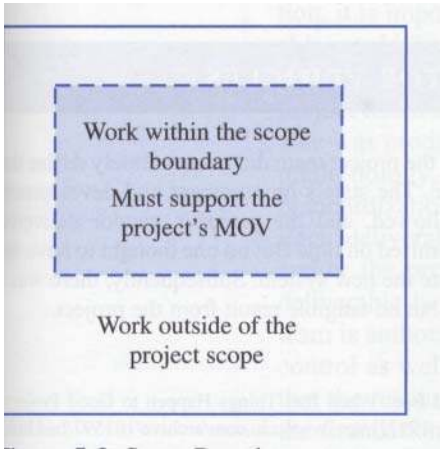

Figure 5.2 Scope Boundary

Scope Statement

1. Develop a proactive electronic commerce strategy that identifies the processes, products, and services to be delivered through the World Wide Web.
2. Develop an application system that supports all of the processes, products, and services identified in the electronic commerce strategy.
3. Integrate the application system with the bank's existing enterprise resource planning system.

It is just as important to clarify what work is not to be included, that is, what work is outside the scope of the project. Often the

scope of a project is defined through interviews, meetings, or brainstorming sessions. Stakeholders often suggest ideas that are interesting, but not feasible or appropriate for the current project.

Let's say that in our example a certain bank vice president pushed for a customer relationship management (CRM) and a data mining component to be included in the application system. The bank's president, however, has decided that the time and effort to add these components cannot be justified because launching the Web site in eight months is vital to bank's competitive strategy. Let's also assume that conducting technology and organizational assessments of our client's current environment is an important piece of our project methodology. But because the bank would like to control some of the costs of this project, we agree that its IT department will conduct that study. The results of this study will then be documented and provided to us.

In this case, it is critical that we define explicitly both what is and what is not part of the project scope. Individuals from both organizations may believe that specific project work (i.e., the assessment study), system features, or functionality (i.e., CRM and data mining) will be part of this project. These beliefs may result in misunderstandings that lead to false expectations or needless work. To manage these expectations, it is useful to list explicitly what is *not* part of the project's scope.

Out of Scope for this Project

1. Technology and organizational assessment of the current environment
2. Customer resource management and data mining components

Setting the scope boundary for the project not only sets expectations, but also can define the constraints of the project and how the product of the organization fits within the organization, that is, the system must integrate with the organization's existing systems.

The scope statement provides a very general and high-level view of the project work and provides only a starting point for defining the scope of our project. At the beginning of a project understanding of the project's scope may be limited. However, as we work more closely with our client more information is uncovered and our understanding of the project increases. Subsequently, the project scope will evolve from being very general and high level to more detailed and defined.

THE OREGON DEPARTMENT OF MOTOR VEHICLES

In 1993, the Oregon Department of Motor Vehicles (DMV) began a project to automate its manual system. The project was originally scheduled to be completed in five years and cost $50 million; but state officials envisioned saving $7.5 million a year by reducing its DMV staff by 20 percent. By 1995, the project's deadline had been extended to 2001 with expected costs escalating to $123 million. A prototype was implemented in a test office in 1996, but soon lines of people backed up around the block. The system was considered a total failure, and the project was cancelled. The project failed because the project team did not accurately define the project's scope. The state's procurement and development rules were followed, and the project's vendor delivered everything promised on time But no one thought to have the vendor integrate the new system. Subsequently, there was a strict process, but no tangible result from the project.

SOURCE: Adapted from When Bad Things Happen to Good Projects, *CIO*, October 15, 1997, http://www.cio.com/archive/101597/bad.html.

PROJECT SCOPE DEFINITION

Developing a scope statement is a useful first step for defining the scope of the project and setting a boundary. A project's scope, however, should also be defined in terms of the deliverables that the team must provide. These deliverables can be divided into project-oriented deliverables and product-oriented deliverables. This separation gives the team a clearer definition of the work to be accomplished and improves the likelihood of accurately assigning resources and estimating the time and cost of completing the work. Moreover, a clear definition of the project's deliverables sets unambiguous expectations and agreement among all of the project stakeholders.

Project-Oriented Scope

Project-oriented deliverables, or scope, support the project management and IT development processes that are defined in the information technology project methodology (ITPM). Project scope includes such things as the business case, project charter, and project plan and defines the work products of the various ITPM phases. Project-oriented deliverables also include specific deliverables such as a current systems study, requirements definition, and the documented design of the information system. These are deliverables supported by the systems development life cycle (SDLC) component of the overall ITPM.

Project-oriented deliverables require time and resources and, therefore, must be part of the overall project schedule and budget. Their role is to ensure that the project processes are being competed so that the project's product (i.e., the information system) achieves the project's MOV and objectives. Project-oriented deliverables also provide tangible evidence of the project's progress (or lack of progress). Finally, they allow the project manager to set a baseline for performance and quality control because they usually require some form of approval before work on the next project phase or deliverable begins.

Project-Oriented Scope Definition Tools All of the project deliverables must have a clear and concise definition. One way to communicate the project's deliverables is to create a **deliverable definition table (DDT).** An example of a DDT for our bank's electronic commerce system is illustrated in Table 5.2.

The purpose of the DDT is to define all of the project-oriented deliverables to be provided by the project team. Each deliverable should have a clear purpose. In addition, it is important to define the structure of the deliverable. For example, a deliverable could be a document (paper or electronic), prototype, presentation, or the application system itself. This sets the expectation of what will be *delivered* by the project team. Moreover, the standards provide a means to verify whether the deliverable was produced correctly. These standards could be defined within the IT Project methodology, controlling agency (e.g., International Organization for Standardization), or through various quality standards established by the organization. Each deliverable must be verified and approved generally by the project sponsor and/or the project manager. It is important that the responsibility for approving a deliverable be clearly defined as well. Once a deliverable is approved, the project team is authorized to begin work on the next deliverable. This provides authorization control as well as a basis for logically sequencing the work. Finally, it is important that the resources required to complete the deliverable be defined. This will provide the foundation for determining not only what resources will be needed for the project, but also for estimating the time and cost in completing each deliverable.

Table 5.2 Deliverable Definition Table

Deliverable	*Structure*	*Standards*	*Approval Needed By*	*Resources Required*
Business case	Document	As defined in the project methodology	Project sponsor	Business case team & office automation (OA) tools
Project charter & project plan	Document	As defined in the project methodology	Project sponsor	Project manager, project sponsor, & OA tools
Technology & organizational assessment	Document	As defined in the project methodology	Project manager & project sponsor	Bank's systems analysts users, case tool, and OA tools
Requirements definition	Document	As defined in the project methodology	Project manager	System analyst, users, case tool, & OA tools
User interface	Prototype	As defined in the user interface guidelines	Project sponsor	System analyst, programmer, users, & integrated development environment (IDE)
Physical & technical design	Document	As defined in the project methodology	Project manager & project sponsor	System analyst, programmer, & case tool
Application system	Files & database	As defined in the project methodology	Project sponsor	Programmers, system analysts, network specialists, program development tools, and relational database management system
Testing plan	Document	As defined in the project methodology	Project manager	System analysts & OA tools
Testing results	Document	As defined in the test plan	Project manager	Programmers, system analysts, & OA tools
Change management and implementation plan	Document	As defined in the project methodology	Project manager	Systems analysts & OA tools
Training program	User documentation & training class	As defined in the implementation plan	Project manager & project sponsor	Trainers, documentation writers, & OA tools
Final report & presentation	Document	As defined in the project methodology	Project sponsor	Project Sponsor, project manager, & OA tools
Project evaluations & lessons learned	Document	As defined in the project methodology	Project manager & senior	Project team, knowledge management system

SOURCE: Inspired by Graham McLeod and Derek Smith, *Managing Information Technology Projects* (San Francisco: Boyd & Fraser, 1996), 51-52.

Once the deliverables have been defined in the DDT, a **deliverable structure chart (DSC)** can be developed as an interim step to define detailed work packages that will be used to estimate the project schedule and budget. Later on, these work packages will be used to create a **work breakdown structure (WBS)**—a tool used to help create the project plan. For example, Figure 5.3 provides an example of a

PROJECT SCOPE: KEEP IT SIMPLE

Since 1994, the Standish Group has studied over twenty-three thousand projects. It found that the number of IT projects delivered on time and within budget for *Fortune* 500 companies increased from 9 percent in 1994 to 24 percent by 1998. The average cost of IT projects has decreased from $2.3 million to $1.2 million as a result of a reduction in project scope. It appears that the likelihood of a project being developed on time and within budget is negatively correlated with project size. In other words, projects that take less than six months, have fewer than six people, and cost less than $750,000 have the highest probability of meeting the schedule and budget objectives. According to Jim Johnson, president of Standish Group International, the best way to design and manage projects is to follow an iterative process that focuses on the most key features. Although more features can be added later on, they will probably be deemed unnecessary. The study also found that user involvement, executive support, experienced project management, clear business objectives, and good communication were important to project success.

SOURCE: Adapted from Kathleen Melymuka, With IT Projects, Small is Beautiful, *Computerworld,* June 18, 1998. http://www.computerworld.com/news/1998/story/0,11280,25731,00.html

Deliverable Structure Chart that maps the project life cycle and systems development life cycle phases to the deliverables defined in the DDT.

Product-Oriented Scope

Although the electronic commerce application system is listed as a project-oriented deliverable, we really do not have any idea what exactly will be delivered to the client. In general, the application system will be the largest project deliverable and will, therefore, require the most time and resources to complete. Identifying the features and functionality of the application system (and their complexity) will be pivotal for estimating the time and cost of producing this deliverable.

Product-Oriented Scope Definition Tools Product scope therefore focuses on identifying the features and functionality of the information system to be implemented.

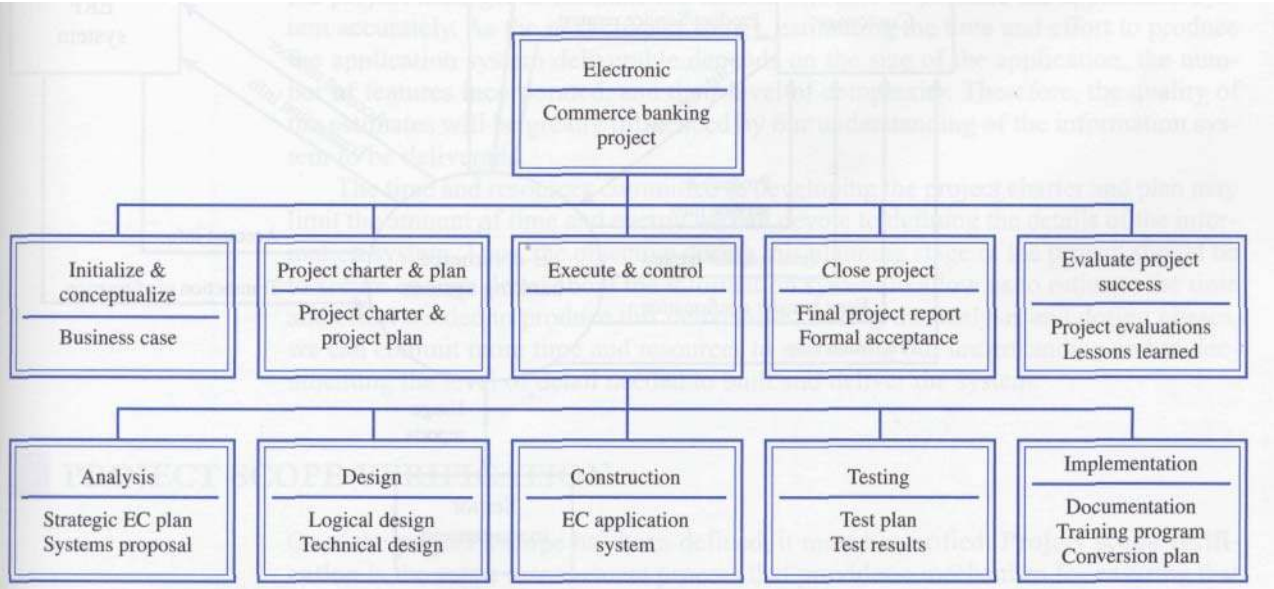

Figure 5.3 Deliverable Structure Chart (DSC)

A useful tool for refining the scope boundary and defining what the system must do is a modeling tool called a context level **data flow diagram (DFD).** A DFD is a process model that has been available for quite some time and is often taught in systems analysis and design courses. A context level DFD, however, presents a high-level representation of the system that has one process (i.e., a circle or rounded rectangle that represents the system as a whole) and depicts all the inflows and outflows of data and information between the system and its external entities. The external entities are usually represented by a square and can be people, departments, or other systems that provide or receive flows of data. Arrows represent the directional flow of data between external entities and the system. Each arrow and entity should be labeled appropriately. Lower level DFDs can be developed later to model the processes and flows of data in greater detail. An example of a context level DFD for our banking electronic commerce system is provided in Figure 5.4. As you can see, the high level features and functionality of the system focus on what the system must do.

Another useful tool for defining the product scope is the **use case diagram,** which has been used in the object-oriented world as part of the Unified Modeling Language (UML). While Jacobson (Jacobson, Cristerson et al. 1992) introduced the use case as a tool for software development, a use case diagram can provide a high level model for defining, verifying, and reaching agreement upon the product scope.

The use case diagram is a relatively simple diagram in terms of symbols and syntax, but it is a powerful tool for identifying the main functions or features of the system and the different users or external systems that interact with the system. At this early stage of the project, the use case can provide a high level diagram that can be further refined and detailed during requirements analysis later in the project.

Actors are people (i.e., users, customers, managers, etc.) or external systems (i.e., the bank's ERP system) that interact, or *use,* the system. Think of actors in terms of roles (e.g., customer) instead of as specific individuals (e.g., Tom Smith). A use case,

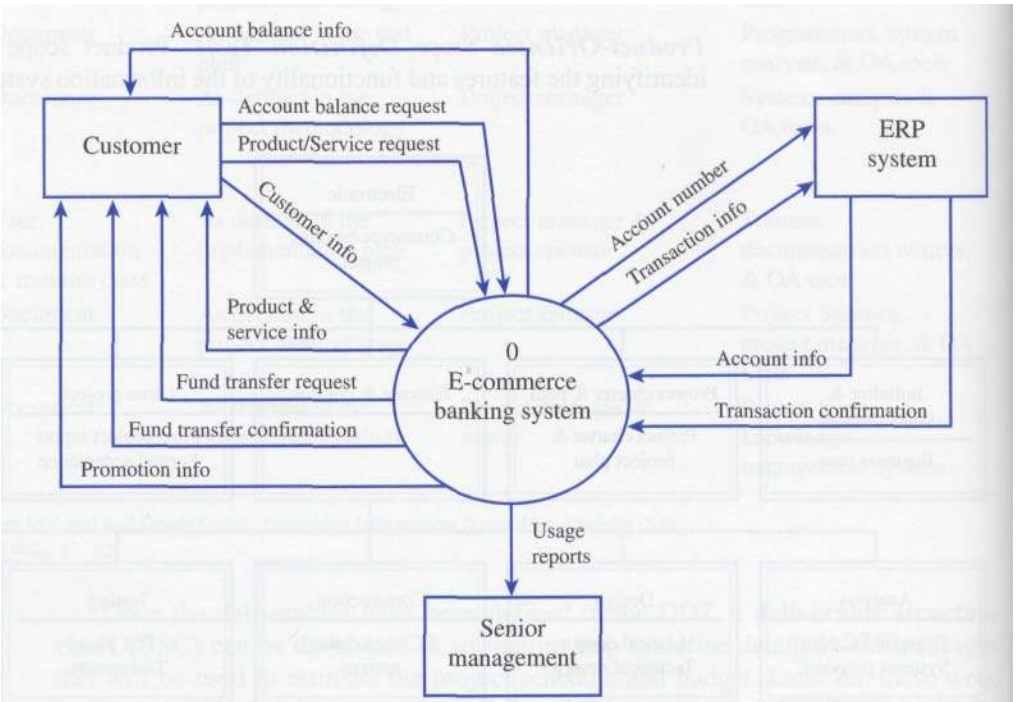

Figure 5.4 Context Level Data Flow Diagram (DFD)

on the other hand, depicts the major functions the system must perform for an actor or actors. When developing a use case diagram, actors are identified using stick figures, while use cases are defined and represented using ovals. Figure 5.5 provides an example of a use case diagram for the bank example.

As you can see in Figure 5.5, the use case diagram provides a simple yet effective overview of the functions and interactions between the use cases and the actors. The box separating the use cases from the actors also provides a system boundary that defines the scope boundary. Use cases inside the boundary are considered within the scope of the project, while anything outside of the boundary is considered outside the scope of the project. Listing the actors provides an opportunity to identify various stakeholders and can be useful for understanding the needs of the organization as a whole. It can be useful not only for addressing competing needs among various stakeholders, but also for identifying security issues as well (Fowler and Scott 1997). The development of a use case diagram is an iterative process that can be developed during a **joint application development (JAD)** session. JAD is a group-based method where the users and systems analysts jointly define the system requirements or design the system (Turban, Rainer and Potter 2001).

The use case diagram used to define the product scope can be used to refine the level of detail and functionality later on in our project. Following our example, the use case diagram in Figure 5.5 identifies the customer actor as using the system to transfer payments. However, a scenario or set of scenarios could be developed during the analysis and design phases of our project to determine how a customer would transfer funds successfully, while another scenario might focus on what happens when a customer has insufficient funds in their account. This level of detail is more suited to the requirements definition rather than the scope definition. At this point, it is more important to identify that the system must allow a customer to transfer funds than to identify how the funds may be transferred. Later on, the product scope can be compared or measured against the detailed requirements. These detailed requirements will be defined during the SDLC component of the ITPM.

But what is the appropriate level of detail for defining the product scope? Knowing the right level of detail is more an art than a science. The right level allows the project manager to estimate the time it will take to produce the application system accurately. As the next chapter shows, estimating the time and effort to produce the application system deliverable depends on the size of the application, the number of features incorporated, and their level of complexity. Therefore, the quality of the estimates will be greatly influenced by our understanding of the information system to be delivered.

The time and resources committed to developing the project charter and plan may limit the amount of time and energy we can devote to defining the details of the information system. Thus, the objective during this planning stage of the project should be to secure enough detail about the information system to allow us to estimate the time and effort needed to produce this deliverable. During the analysis and design phases, we can commit more time and resources to increasing our understanding and to documenting the level of detail needed to built and deliver the system.

PROJECT SCOPE VERIFICATION

Once the project's scope has been defined, it must be verified. **Project scope verification** is the scope management process that provides a mechanism for ensuring that the project deliverables are completed according to the standards described in the

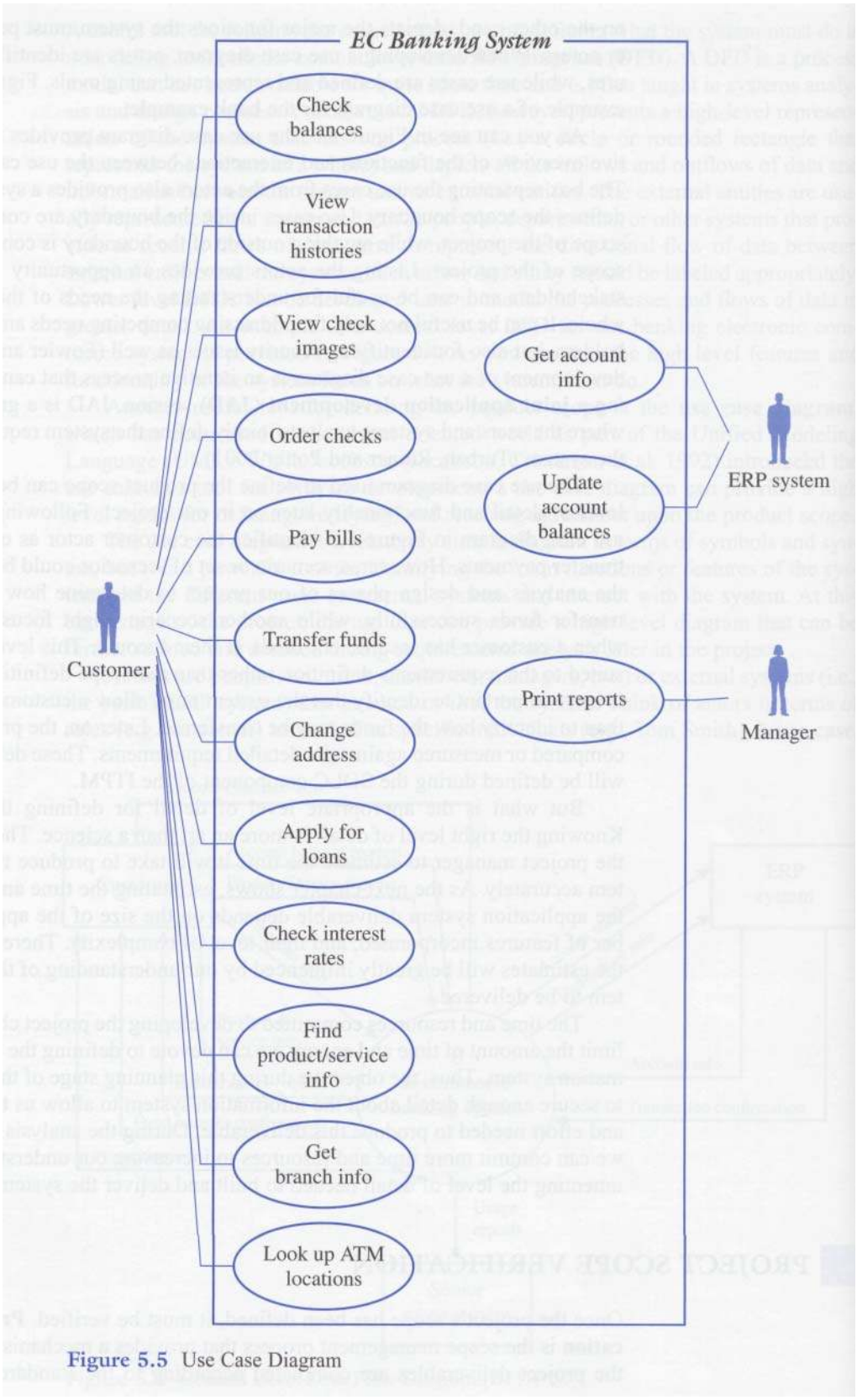

Figure 5.5 Use Case Diagram

DDT. Gray and Larson (2000) provide a project scope checklist for ensuring that the deliverables are completed—and completed correctly. This checklist has been adapted to include the MOV concept.

- *MOV*—Are the project's MOV clearly defined and agreed upon? Failure to define and agree upon the MOV could result in scope changes later in the proj ect, which can lead to added work impacting the project's schedule and budget.
- *Deliverables*—Are the deliverables tangible and verifiable? Do they support the project's MOV?
- *Quality standards*—Are controls in place to ensure that the work was not only completed, but also completed to meet specific standards?
- *Milestones*—Are milestones defined for each deliverable? Milestones are significant events that mark the acceptance of a deliverable and give the project manager and team the approval to begin working on the next deliv erable. In short, milestones tell us that a deliverable was not only com pleted, but also reviewed and accepted.
- *Review and acceptance*—Are both sides clear in their expectations? The project's scope must be reviewed and accepted by the project stakeholders. The project sponsor must formally accept the boundary, product to be pro duced, and the project-related deliverables. The project team must be clear on what it must deliver. In both cases, expectations must be realistic and agreed upon.

SCOPE CHANGE CONTROL

According to the PMBOK, **scope change control** is concerned with ensuring that any changes to the project scope will be beneficial, with determining that an actual scope change has occurred, and with managing the actual changes when and as they occur. Scope control is also concerned with:

- *Scope grope*—Scope grope is a metaphor that describes a project team's inability to define the project's scope. This situation is common early in a project when the project team and sponsor have trouble understanding what the project is supposed to accomplish. Scope grope can be minimized by having a clearly defined MOV and by following or applying the processes, concepts, and tools described in this chapter.
- *Scope creep*—Scope creep refers to *increasing featurism*, adding small yet time- and resource-consuming features to the system once the scope of the project has been approved. For example, a project sponsor may try to add various bells and whistles to the project scope. Yet, scope creep does not always come from the project sponsor side. The project team itself may come across interesting or novel ideas as the project work progresses. Its enthusi asm for adding these ideas can divert its attention or add features and func tions to the system that the project sponsor did not ask for and does not need. Scope creep must be identified and controlled throughout the project because it will lengthen the project schedule and, in turn, lead to cost overruns.
- *Scope leap*—If scope creep is caused by increasing featurism, scope leap suggests a fundamental and significant change in the project scope. For example, the original scope for the bank's electronic commerce project was

SIX MYTHS OF SCOPE MANAGEMENT

Myth 1: User involvement will result in an IS project grounded in the realities of business needs.

Reality: Often user involvement is really a vaguely stated idea from senior management handed off to someone in the user community. Involvement by proxy can create problems if the original *concept person* is too busy or unavailable to discuss the details.

Myth 2: A scope statement will clearly define what a project will do.

Reality: A good scope statement will also make it clear as to what the project will not attempt to do, which is especially important when specifying roles and responsibilities. Setting scope is much like putting a fence around the project. It not only keeps things in, it also keeps things out.

Myth 3: Once the scope of the project is defined, hold firm because any deviation from the original plan is a sign that the project is out of control.

Reality: Scope change is inevitable. Often schedules and budgets are set before enough details of the project are known. Early estimates should be revised as new information is acquired. Good scope management, however, involves a change management committee of senior management who review proposed changes and decide whether an additional feature or requirement should be added to the project's scope.

Myth 4: A function of a scope change committee is to arbitrate user requests for additional features or functionality beyond the original project charter.

Reality: Scope problems go beyond additional user demands. Scope changes will affect schedule, budget, or both. Slippage of the schedule will require additional resources or reduced functionality. It is also important that the project not get off track while the scope change committee reviews a particular change.

Myth 5: Regular and frequent meetings with senior management will ensure they are kept up to date and will result in goodwill and support.

Reality: They may not be listening. It is important to keep their attention and involvement by focusing on the benefits of the system.

Myth 6: You can always make up schedules and budgets later on if they slip a little bit.

Reality: Catching up is a rare occurrence. Projects rarely fail overnight, and project managers must be vigilant for early warning signs. If there are minor setbacks, it is important that the project manager be candid with senior management.

SOURCE: Adapted from Alice LaPlante, Scope Grope, *Computerworld*, March 20, 1995, http://www.computerworld.com/news/1995/story/0,11280,1340,00.html.

to provide new products and services to its customers. Scope creep may be adding a new feature, such as a new product or service, not originally defined in the project's scope. Scope leap, on the other hand, is an impetus to change the project so that the electronic commerce system would allow the bank to obtain additional funding in the open market. Adding this activity would dramatically change the entire scope and focus of the project. Scope leap can occur as a result of changes in the environment, the business, and the competitive makeup of the industry. Scope leap entails changing the MOV and, therefore, requires that the organization rethink the value of the current project. If this change is critical, the organization may be better off pulling the plug on the current project and starting over by conceptualizing and initiating a new project.

Scope Change Control Procedures

A scope change procedure should be in place before the actual work on the project commences. It can be part of, or at least referenced in, the project charter so that it is communicated to all project stakeholders. This procedure should allow for the identification and handling of all requested changes to the project's scope. Scope change requests can be made, and each request's impact on the project can be assessed. Then, a decision whether to accept or reject the scope change can be made.

JUST ONE MORE BELL—ONE MORE WHISTLE

Scope creep is a widespread problem. A *Computerworld* survey of 160 IS professionals revealed that 80 percent of the respondents said scope creep "always" or "frequently" occurs. Moreover, 44 percent responded that "poor initial requirements definition" was the leading cause for scope creep. In addition, only 16 percent of the respondents said "no" when users demanded significant changes once the project was well under way. To reduce scope creep, 63 percent of the respondents use JAD and 25 percent use prototyping. A classic case of scope creep is provided in a research study by Mark Keil of Georgia State University, which focused on an artificial intelligence application designed to help sales people of a large computer vendor configure computer systems. Although scope creep may arise as a result of not getting the scope of a project defined properly, Keil found that sales staff did not care for the system because they, the system's would-be users, were rewarded on sales volume and not the accuracy or completeness of the order. Thus, they began to make excuses for not using the system. These excuses became scope changes, and the project began to take on a life of its own that led to project escalation. The company eventually canceled the project after the company spent tens of millions of dollars over eleven years.

SOURCE: Adapted from Mark Keil, Pulling the Plug: Software Project Management and the Problem of Project Escalation, *MIS Quarterly,* December 1995, 421–447; and Gary H. Anthes, No More Creeps: Are You a Victim of Creeping User Requirements? *Computerworld,* May 2, 1994, http://www.computerworld.com/news/1994/story/0,11280,15919,00.html.

A scope change procedure may include a scope change request form. An example of a scope change request form is illustrated in Figure 5.6. The individual or group making the scope change request should complete the form.

Regardless of the format for a scope change request form, it should contain some basic information. First, the description of the change request should be clearly defined so that the project manager and project team understand fully the nature and reason for the scope change. Second, the scope change should be justified, which separates the *would likes* from the *must haves.* In addition, several alternatives may be listed in order to assess the impact on scope, schedule, resources, and cost. Often a trade-off or compromise will be suitable if the impact of the scope change is too great. The project sponsor must understand and approve these impacts because the baseline project plan will have to be adjusted accordingly. Alternatives may include reducing functionality in other areas of the project, extending the project deadline, or adding more resources in terms of staff, overtime, or technology. Finally, all scope changes must be approved so that additional resources can be committed to the project.

However, nothing can be more frustrating than making a request and then not hearing anything. Too often requests fall through the cracks, leading to credibility concerns and accusations that the project manager or project team is not being responsive to the client's needs. Therefore, a scope change control procedure should be logged with the intention that each request will be reviewed and acted upon. As seen in Figure 5.7, an example of a Change Request Log includes information as to who has the authority to make the scope change decision and when a response can be expected.

Although this may seem like the beginning of a bureaucracy, it is really designed to protect all project stakeholders. Too often the project manager and project team feel the pressure to say yes to each and every scope change request because their refusal may be interpreted as being uncooperative. Unfortunately, introducing scope creep will impact the schedule and budget. As the deadline passes or as costs begin to overrun the budget, the project manager and team then may come under fire for not controlling the project objectives.

Scope change request form

Requestor name: _____ Request date: _____

Request title: _____ Request number: _____

Request description:

Justification:

Possible alternatives:

Impacts	*Alternative 1*	*Alternative 2*	*Alternative 3*
Scope			
Schedule			
Resources required			
Cost			

Recommendation:

Authorized by ***Date***

Figure 5.6 Scope Change Request Form

Request Number	*Request Title*	*Date of Request*	*Requested By*	*Priority (L, M, H)*	*Authority to Approve Request*	*Expected Response Date*	*Scope Change Approved? (Y/N)*

Figure 5.7 Scope Change Request Log

Still, a project manager and team should not say no to every scope change request. Some changes will be beneficial and warranted as the project proceeds. The question then becomes, What should be the basis for making a scope change decision?

As you have seen, the project's MOV guides the project planning process. Similarly, the project's MOV can also guide scope change decisions. A scope change request should be approved if—and only if—the scope change can bring the project closer to achieving its MOV; otherwise, why bother adding additional work, resources, time, and money to activities that will not bring any value to the organization?

Benefits of Scope Control

The most important benefit of scope change control procedures is that they keep the project manager in control of the project. More specifically, they allow the project manager to manage and control the project's schedule and budget. Scope control procedures also allow the project team to stay focused and on track in terms of meeting its milestones because it does not have to perform unnecessary work.

CHAPTER SUMMARY

Although scope is the work to be performed on the project, a project's scope can be defined as the boundary and deliverables that the project team will provide to the project sponsor. A scope boundary acts as a fence to ensure that what needs to get done, gets done—and only what needs to get done, gets done. Performing work that does not help the project achieve its MOV needlessly consumes valuable time and resources. Therefore, the project's boundary helps the project team define the limits of the project and how it will interact with its environment. In addition, deliverables are tangible units of work that ensure that the project is on track. Deliverables may be product-oriented or project-oriented. Product-oriented deliverables focus on the high level features and functionality of the application system—the project's product. On the other hand, project-oriented deliverables focus on the project's processes as defined in the IT project methodology.

The Project Management Body of Knowledge identifies five processes that make up the scope management process group. These processes include: (1) Scope Initiation, (2) Scope Planning, (3) Scope Definition, (4) Scope Verification, and (5) Scope Change Control. Figure 5.8 summarizes these processes and the tools used to support them.

Scope grope is a common occurrence in the early stages of the project. Often the project team struggles to define what the project is all about and what work must be done. By applying the concept of an MOV and the tools introduced in this chapter, the time a project team spends searching for these answers should be reduced. Scope creep, on the other hand, is a common occurrence in many projects. It entails adding additional features or functions to the scope once the scope has been set and approved. This phenomenon can increase the schedule and budget, causing the project to miss its deadline and budget targets. Scope creep can be managed by (1) verifying that the scope is accurate and complete by using a scope verification checklist, and (2) ensuring that appropriate scope changes are approved and reflected in the baseline plan by having scope change procedures. The MOV concept can guide this decision process. For example, scope changes that move the project closer to achieving its MOV should be approved, while those that do not merely waste time and resources. Lastly, scope leap entails a major and fundamental change to the project scope. It may be the result of a changing business environment or the competitive makeup of the industry. Such a radical departure from the original business case may require the project stakeholders to rethink the viability of the current project.

REVIEW QUESTIONS

1. What is meant by project scope?
2. Briefly describe the five scope management processes.
3. What is the project's scope initiation process? When does it occur? Why is it important?
4. How is a project's scope initiation process supported by the IT project methodology?
5. Briefly describe the scope planning process.
6. Briefly describe the scope definition process.
7. Briefly describe the scope verification process.

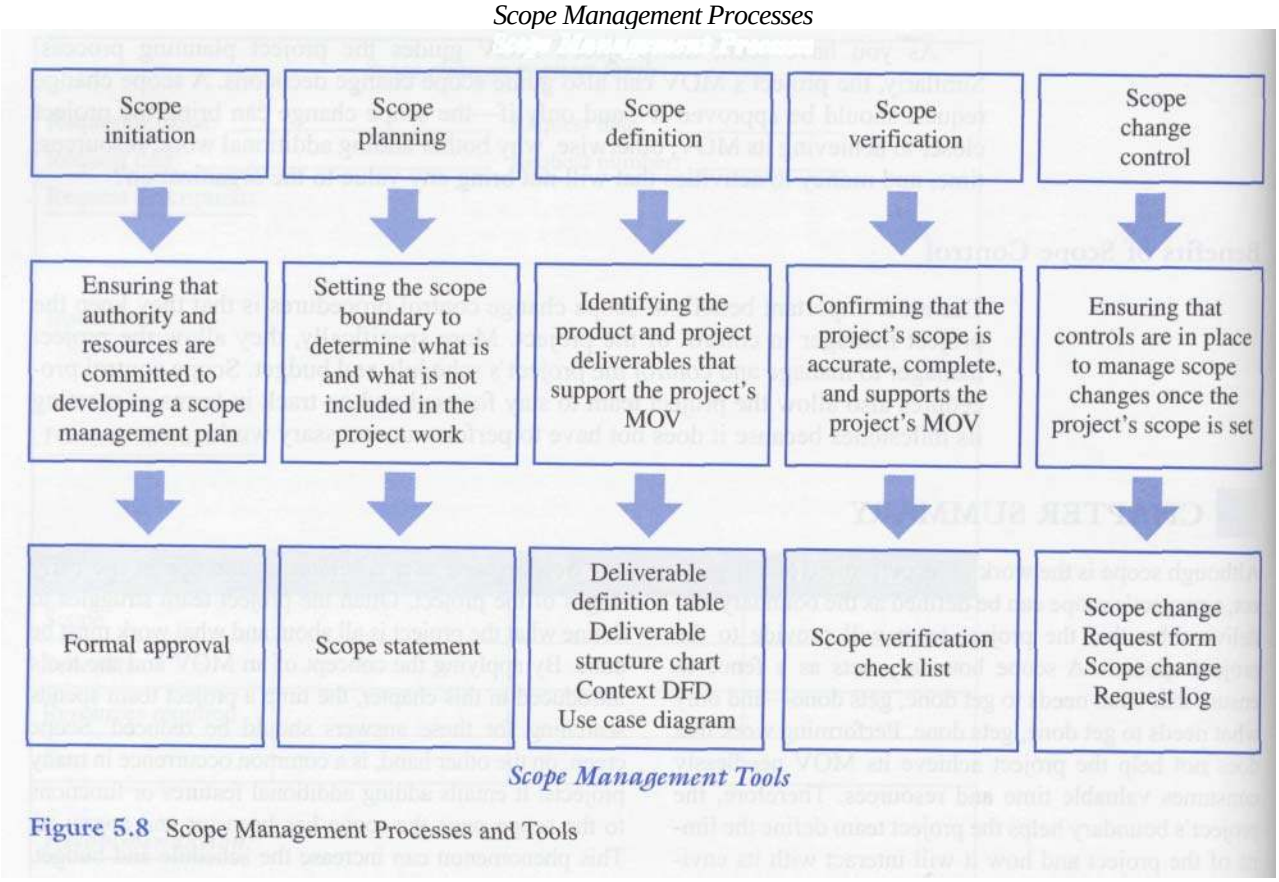

Figure 5.8 Scope Management Processes and Tools

8. Briefly describe the scope change control process.
9. Describe the scope management plan in Figure 5.1.
10. Why is it important to define the project's scope accurately and completely?
11. What is a scope boundary? What purpose does it serve?
12. What is the difference between product-oriented deliverables and project-oriented deliverables?
13. How does a project's scope support the MOV concept?
14. What is a scope statement? What purpose does it serve?
15. What is a context dataflow diagram (DFD)? What purpose does it serve?
16. How does a use case diagram help to define the pro ject's scope?
17. What is a deliverable definition table (DDT)? What purpose does it serve?
18. What is a deliverable structure chart (DSC)? How does it map to a deliverable definition table (DDT)?
19. What is a work breakdown structure (WBS)? How does it map to the DDT and DSC?
20. Briefly describe what must be included in a scope verification checklist?
21. What is the purpose of verifying a project's scope?
22. What is the purpose of scope change control pro cedures?
23. Briefly describe scope grope.
24. Briefly describe scope creep.
25. Briefly describe scope leap.
26. What are the benefits of having scope control pro cedures?
27. Briefly describe what should be included on a scope change request form.
28. What is the purpose of a scope change request log?

EXTEND YOUR KNOWLEDGE

1. Using the Web or library, find an article about an unsuccessful IT project. Discuss whether poor scope management had any bearing on the project being unsuccessful.
2. Discuss the statement: Failing to define what is not part of the project is just as important as failing to define what is part of the project.
3. Choose a company that sells a product or service on the Web. Using this Web site as a guide, develop the following (even though the application is already in existence). You may make assumptions where necessary, but be sure to document them.
   a. Scope statement
   b. Context level DFD
   c. Use case diagram

BIBLIOGRAPHY

Fowler, M. and K. Scott 1997. UML Distilled: Applying the Standard Object Modeling Language. Reading, Mass.: Addison-Wesley.

Gray, C. F. and E. W. Larson 2000. Project Management: The Managerial Process. Boston: Irwin McGraw-Hill.

Jacobson, I., M. Cristerson, P. Jonsson, and G. Overgaard 1992. Object-Oriented Software Engineering: A Use-Case Driven Approach. Reading, Mass.: Addison-Wesley.

Turban, E., R. K. Rainer, Jr., and R.E. Potter 2001. Introduction to Information Technology. New York: John Wiley.