PART one

Overview

CHAPTER 1

Introduction

THE rest of this book is divided into five parts, corresponding to the four tasks that the project manager performs—planning, organizing, implementing, and measuring—plus a fifth part on case studies. For architecture-centric software project management, the project manager works closely with a chief architect and uses and contributes to many of the artifacts that result from high-level architecture design. This chapter gives an overview of all the main activities that a project manager practicing an architecture-centric approach will be involved in.

1.1 What Is Project Management?

Project managers are involved in four primary activities:

- Planning
- Organizing
- Implementing
- Measuring

For software management, although these activities overlap a lot, they follow somewhat the sequence in which I present them. Projects are usually *planned* at their beginning. I envision a high-level design phase of a project, when much of the project planning is done while the architecture is being designed.

Planning will also occur again prior to every increment of development. I prefer to keep the increments shorter than eight weeks in duration.

Organizing is also done early in a project. The project team organization needs to be set up and the roles of the team members defined. In some cases, a project manager may be responsible for recruiting and selecting the members of the project team. In other cases, technical staff may be assigned to the project, and the project manager must then allocate the development team roles.

The project manager is responsible for *implementing* the project in accordance with the plans that have been developed. Often, the project manager will need to react to unforeseen situations, since plans are rarely implemented exactly as conceived earlier in the project. How the project manager reacts to these situations, the decisions that are made, the replanning that is done, and so on will affect the outcome of the project.

The project manager must *measure* the progress of the project and the characteristics of the product as it is being developed. He or she will also measure the performance of the project team, its members, and the effectiveness of the product they produced, both during and after the project is completed.

There are obviously many other tasks a project manager must be involved in, such as leading, controlling, setting customer expectations, innovating, deciding, directing, and mentoring, Some of these are described in more detail in Chapter 8. The **planning, organizing, implementing, and measuring (POIM)** definition of what a project manager does is simple to remember, and it functions as a convenient summary of major project management tasks.

How you spend your time will determine to a large extent how successful you are as a project manager. There is never enough time to get everything done, and you will need to carefully determine priorities among tasks to provide a successful balance. For example, in my experience, developing quarterly status reports has low priority, since very few people take time to read them. Their value is often limited, since you are reading about something in the past that you cannot affect. However, though writing quarterly reports is a requirement for many project managers, I myself try to spend as little time as possible writing them. Recognize the important action items, do those first, and push other, lower-priority tasks to the bottom of your personal queue. You will also have to balance the mix among the actions required of you by your development project and those demanded by your personal (non-work-related) life.

Good software project managers usually have a balanced mix of technical and people-handling skills. People management skills are often characterized in terms of communication, vision, leadership, empathy, teaching, charisma, and so on. These skills are often harder to learn than are technical skills. This book may not make you a better communicator, for example, but it can provide tips as to what types of information are important to communicate to the development team. Technical skills can often be augmented by a strong chief architect. Thus, people management skills are somewhat of a prerequisite for software project managers. Many of these skills have nothing to do with software architecture. They can also be developed outside of the project team and the work environment. I encourage new software project managers to build their people management skills at every possible opportunity. This could be done in other environments, such as while managing the company softball team or organizing an external speaker series, with less risk than in development projects.

1.2 What Is Software Architecture?

I will rely on the definition of software architecture found in Soni, Nord, and Hofmeister [1995]:

> Software architecture is concerned with capturing the structures of a system and the relationships among the elements. . . . The structures we found fell into several broad categories: conceptual architecture, module architecture, execution architecture, and code architecture.

The software architecture of a product serves both as a design plan and as an abstraction of the product envisioned to be implemented. I have seen within the past ten years a growing awareness of the importance of describing software architecture before implementing a product. This relates directly to software project management, since the project plan must support the creation and description of the architecture. Once created, the architecture must be described to every member of the development team. Quite simply, I have found that development teams with a good vision of what they are implementing have a better chance to successfully implement the product. For architecture-centric software project management, the project manager recognizes and supports

the realization that a good software architecture is necessary for developing a good product.

1.3 Core Beliefs

The project management practices described in this book can be characterized as "middleweight" processes. This is somewhere between heavyweight processes such as those described within the **capability maturity model (CMM)** [Paulk 1995] and the **Rational Unified Process (RUP)** [Kruchten 1999; Royce 1998] and lightweight processes such as **extreme programming** [Beck 2000]. Basically I believe you should spend some time up front designing a software architecture for the envisioned product as well as planning the project. But use incremental development to get to the market quickly while incrementally updating your plan as you implement the architecture. These middleweight processes have the following fundamental characteristics.

- *Architecture design and description:* I believe that successful software development projects need a description of the architecture design that can be understood by all development team members. The architecture design should be done prior to writing large amounts of production code.
- *Project planning:* Projects should be planned while the software architecture is being designed. Schedules, effort estimates, and the project organization should be based on the architecture. Schedules developed prior to having an architecture design are likely to be very inaccurate.
- *Incremental development:* Software products should be developed incrementally, based on the software architecture and the set of desired features. The first increment, which I call a "vertical slice," should be used to prove major characteristics of the architecture.
- *Project manager/architect team:* The project manager and software architect should work closely together as a decision-making team. Responsibilities are roughly divided between management and technical decisions. Successful projects consisting of more than a few developers should have two different individuals in these roles.
- *Trade-off analysis:* Project management consists of a set of trade-off decisions. Thus, there are no right or wrong answers, and projects that

are implemented will never go exactly as planned. Project managers must be flexible to be able to best manage development risks.

- *Soft factors:* Soft factors will affect a project as much as or more than the technical issues. These soft factors include team building, morale, management influences, business influences, staff experience, and culture.

1.4 Project Management Process

The process a project manager uses to do his or her job will depend greatly on the software development process of the larger organization. It is *not* my intent to describe a rigorous procedure that every project manager should follow. Rather, to more easily introduce the book's concepts, I identify many of the steps the project manager is likely to follow when using an architecture-centric approach. This will help set the context for the practices described in the book and introduce what the project manager does and how that relates to architecture design activities.

Typically, a development process or work flow is set up as a sequence of steps with well-defined inputs and outputs along with roles and responsibilities. Project management is to a large extent an iterative process. Plans are formulated and an attempt made to follow them. But unexpected events arise, and plans must frequently be modified. These plan modifications are called *mid-course corrections*. Also, project managers sometimes make decisions based on partial information. Often, a straw-man plan will be made only to be refined or replaced as more information becomes available. Thus, I will not attempt to define precise processes or work flows for project managers. But I will attempt to list many of the steps the project manager should consider, in a rough sequence, as he or she attempts to plan, organize, implement, and measure a project working closely with the chief architect.

1.5 Architecture-Centric Project Management

I provide here an overview of how you could manage a software project that has architecture design at the center. Again, the specific sequence of tasks will depend on your software development process. But identifying the advantages and useful outputs of architecture design will help put the tips contained in

later chapters of this book into a framework. You will perform many tasks as a project manager that have nothing to do with the design. But, very simply, over the course of managing a project you will need to make certain that you have an architecture design that represents the vision of the product that will be developed. Your software development plan will identify how you will achieve the vision that the architecture represents. As you manage the development, you will be guided by your plan as well as your progress in achieving the architecture. You will need to make many mid-course corrections to your development plan, but hopefully the architecture will remain more constant while it is being implemented.

Figure 1.1 identifies some of the major steps associated with implementing an entire project using an architecture-centric approach. The boxes represent the activities that will generally be done during the project, and the arrows represent some of the outputs that result from the activities and are inputs to other activities. This project management book will cover the period from when market requirements have been defined to when the software is delivered to the first customer. Thus, it does not focus on project management prac-

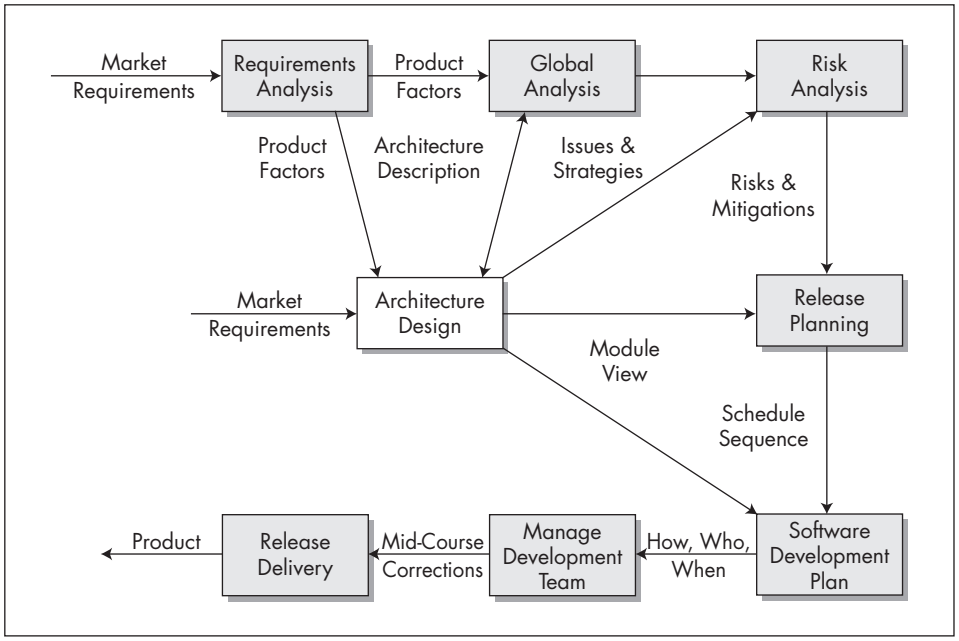

Figure 1.1 *Architecture-Centric Project.*

tices for software products that are mostly being maintained rather than developed. The following sections introduce these major and other steps in the project management activities of planning, organizing, implementing, and measuring.

Architecture-centric project management starts when an initial set of market requirements has been defined. The project manager and the architecture design team analyze the requirements. They analyze product factors and other influences as part of a global analysis. They analyze risks as part of the risk analysis. The module view of the architecture along with the feature list resulting from the requirements analysis become the basis for the release planning for the incremental development.

Once release planning is complete, a software development plan can be developed that also includes the schedule, estimated development costs, and project organization. The **software development plan (SDP)** is generally written for the entire project, with more detailed plans and specific tasks for the first incremental release. This document will be revised for each release increment as the build plan is developed in detail for each increment. I suggest keeping the increment duration small, to within eight weeks between releases. Once development starts, beginning with the detailed design of each software component identified within the module view of the architecture, you will be leading and managing the development team. You will use the schedule and organization identified in the software development plan as a guide. Mid-course corrections to the development plan are inevitable, since not everything will go as planned and unexpected events will occur (e.g., illness of a team member).

At the end of the development increment, the software will be released on the date agreed to in the release plan. You will then plan and implement a cycle of increments until the product functionality is sufficient for an initial product. This initial product will be delivered to users, possibly at first within some type of beta testing or for a lead customer. Within release delivery, you will review test results and data, in order to determine if the product quality is adequate for customer delivery. After the first increment is released, you will likely use a system-testing function to perform independent testing on the software while your development team is working on the next increment. You will plan the project so that you have test results available from the prior increment before

you release the next increment. This helps improve product quality over time and avoids having to do all the testing and fixing at the end of the project.

1.6 Planning

The major steps involved with planning an architecture-centric software project are summarized in Figure 1.2.

Project planning starts when a first set of requirements is defined and ends with the software development plan. Project planning happens in parallel with the design of the software architecture in what I call the high-level design phase of the project. It will also happen before the development of each incremental release. The project manager works with and manages a small architecture design team during the high-level design phase. The chief architect is the technical leader of the design team. The project manager, along with the design team, contributes to the global analysis. The project manager focuses more on

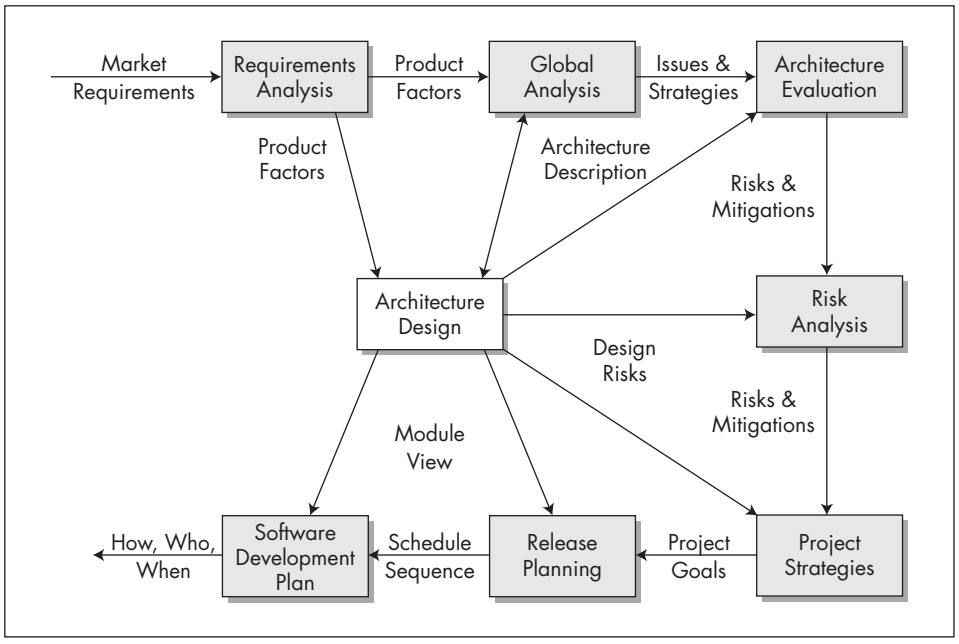

Figure 1.2 *Project Planning.*

organizational influencing factors, while the architects focus more on product and technological factors.

The project manager starts developing top-down schedule estimates in order to get a feel for the scope of the development that will be required to implement the architecture. Once the module view of the software architecture is defined, you can begin putting together bottom-up estimates for all the components of the architecture that will need to be developed. If possible, the members of the anticipated development team will develop the bottom-up estimates. When the architecture is described enough, you will schedule an architecture evaluation review, using both internal and external reviewers. You will keep track of the progress of the high-level design phase so that it is completed in a time frame consistent with the rules of thumb I suggest in Chapter 2.

As project manager you will begin release planning, based on the set of requirements, the bottom-up estimates, and the project milestones that should be reached. You will describe release feature content and define engineering releases and incremental builds within a build plan. You will be making key decisions with the chief architect concerning the technologies needed to implement the architecture.

Global analysis, the architecture design, and the architecture evaluation will help to identify the risks associated with implementing the architecture. As project manager you will further analyze these risks, propose possible risk mitigation actions, and identify project goals and strategies.

In addition to the architecture design specification, the other primary artifact resulting from the high-level design phase is the software development plan (SDP). The SDP contains the schedules and staffing plans for implementing the product. You will update the SDP for each successive release increment. The SDP will identify risk mitigation actions as project tasks. The SDP information is typically used for a management gate review at the end of the high-level design phase, prior to the beginning of development. Management is interested in learning how long the development will take and how much it will cost as an engineering investment. Depending on the results of the review, you may need to modify the SDP, which may also mean changing the architecture design. For example, if the required size of the proposed development team cannot be staffed, the SDP may need to extend the schedule or the architecture may need to be simplified to require less implementation. Clearly, you

could also simplify the requirements. Project planning is a set of trade-offs that hopefully will result in a plan that can be staffed within an acceptable schedule, set of features, and quality level.

1.7 Organizing

The project manager will need to organize the architecture design team, the development team, and all the activities associated with project management. Project management activities include the interfaces to other functions within the organization, such as marketing, quality assurance, system testing, and documentation development. The major steps involved with organizing an architecture-centric software project are summarized in Figure 1.3.

At the at the beginning of the high-level design phase, you will need to establish the architecture design team and the chief architect. The development team members will first get involved with the project during the bottom-up estimation. Ideally, the engineer who is to implement a specific component prepares the estimate for that component. Thus, the project manager needs to

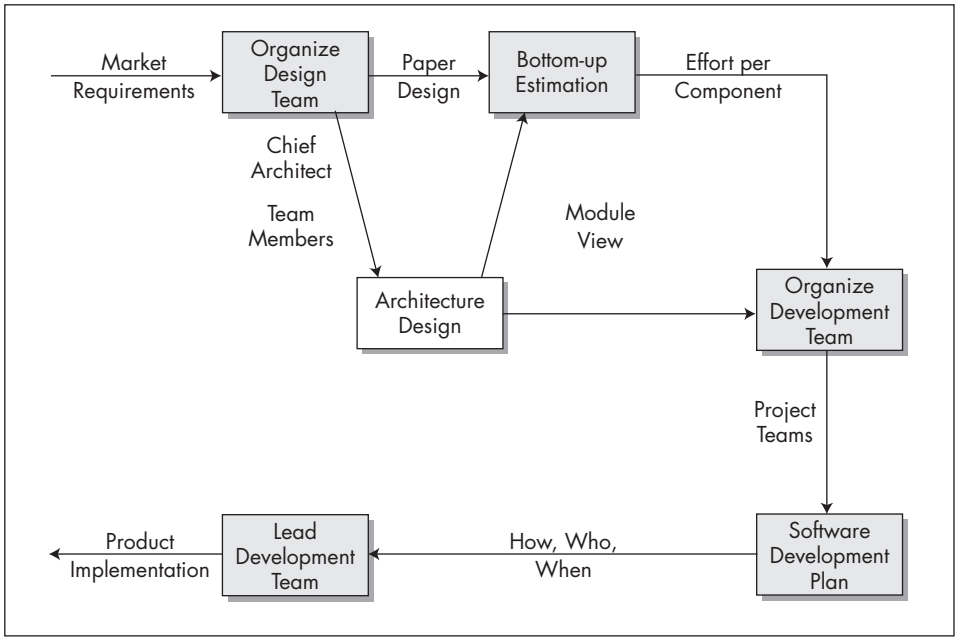

Figure 1.3 *Project Organizing.*

anticipate who the development team members will be and what their roles will be. Each team member will be given the architecture design specification and will have up to four hours to create a "paper design" and fill out a cost estimation form for each component he or she will be estimating.

The project manager will then start organizing the development team. Ideally, the architecture team members will become team leaders, and they will lead the development of each subsystem described in the architecture. Thus, the development team organization should look like the module architecture developed during high-level design.

The SDP will propose the organization, along with task assignments, schedule, and the definitions of the project team role. Most likely, at the end of the high-level design phase, you will hold a management review to determine the availability of each individual that the SDP identifies as needed to implement the product. In cases when key people are unavailable, you may need to modify the SDP or the architecture to accommodate the staffing plan.

After implementation begins, you, as project manager, will be the primary leader of the product development organization. You will need to refine the organization as the product is being developed. This may mean reassigning team members to specific tasks as they are being implemented. In general, critical tasks should be covered by the more experienced engineers. Engineers struggling with achieving their tasks may need support from a mentor or a more experienced engineer, or they may even be removed from the team if their skills or work habits are inconsistent with the needs of the project.

You will have a team of key contributors on whom you should be able to rely to implement the details of the project and make decisions. Foremost within this organization will be the chief architect, who will address many of the technical issues. Depending on the overall size of the project, you will also rely on the team leaders, who will direct the work of the engineers assigned to implement the major subsystems.

1.8 Implementing

The project manager is responsible for implementing the project in accordance with the software development plan. Often, you will need to react to unforeseen situations, since plans are rarely implemented as they were conceived earlier in

the project. How you react to these situations, the decisions you make, what you replan, and so on will affect the outcome of the project. The major steps involved with implementing an architecture-centric software project are summarized in Figure 1.4.

The module view that comes from the architecture design provides inputs for organizing the development team. Risk analysis produces mitigation actions that are used to do the release planning, that is, the sequence in which the components and features will be implemented. The build plan, along with the effort estimates for each component, gives some of the inputs for creating the software development plan. The project manager implements the project in accordance with the development plan. Progress is monitored via weekly status meetings and other communications with team members. As the project is implemented, mid-course corrections will be made to meet the release dates. Release delivery will plan the details of each software release, including its functionality and quality. Each incremental release will be delivered to a system testing function. Later releases will be delivered as a product to customers from the time that a minimal useful set of features is implemented.

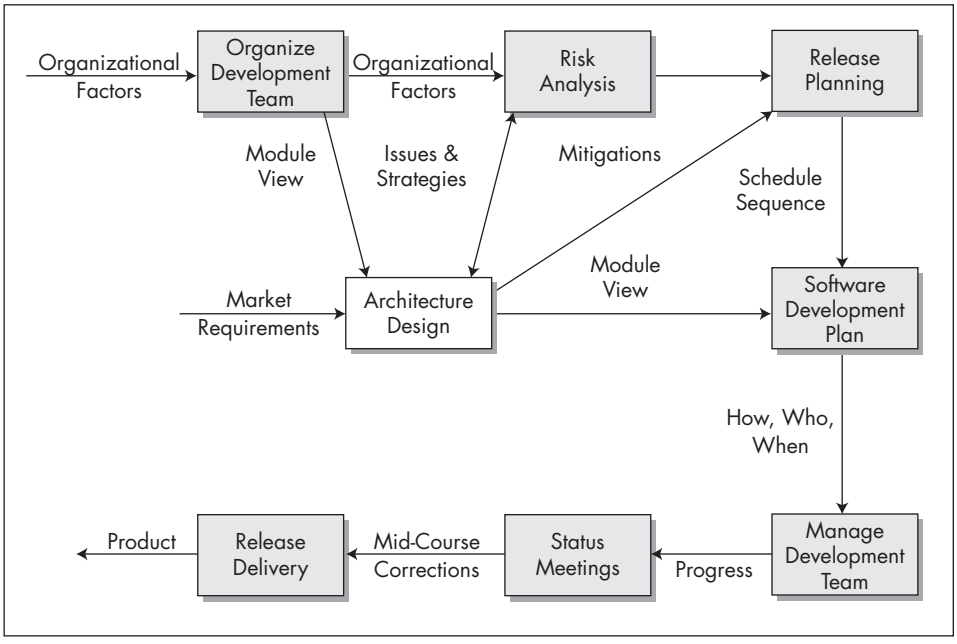

Figure 1.4 *Project Implementing.*

1.9 Measuring

The major steps involved with measuring an architecture-centric software project are summarized in Figure 1.5.

The architecture design will serve to drive the project planning to the point where a proposed software development plan can be written. From the software development plan, a number of metrics will be defined as target goals for the project. These target goals will include measures such as the development budget as calculated from the staffing plan, key milestone dates from the development schedule, size measures, and quality measures. These measures should be consistent with the overall project goals. Managing the project will involve making trade-offs among these goals, and the measures will yield insights with respect to achieving the goals. The goals and measures will likely require trade-offs among meeting the time schedule, providing desired functionality, achieving the quality of the resulting release, and the development costs.

As the software is developed, progress will be made and containing the measures tracked. An important measure for the overall project progress will be whether or not the incremental release dates are being met. After a release

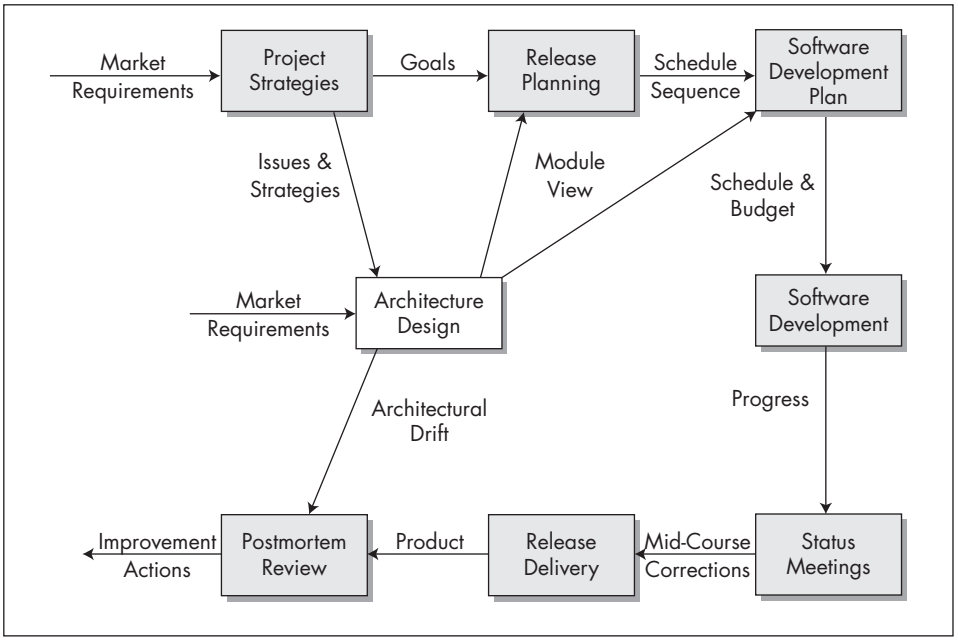

Figure 1.5 *Project Measuring.*

containing the last planned major set of features, you will likely perform a postmortem review on your project. Here you will compare the goals you set during planning with the measures tracked during development. The review will give you insights into how well the project went—such as whether you met your schedule, budget, quality, and functionality goals—and how you can better plan and manage your next project.

1.10 Summary

This chapter provides a broad overview of the major steps involved with architecture-centric software project management. The following chapters describe in more detail what the project manager can do and supply tips for achieving the project management activities.