Chapter 4
Project Management Essentials

- Project management is an involved and complicated set of tasks. Several tasks that will impact you as a developer the most:
  - Project planning
  - Estimation and scheduling
  - Resource management
  - Project oversight
  - Project reviews and presentations
  - The project retrospective

Project Planning

- Project planning is ***forever***.

- **What's in the project plan? Generally a project plan consists of the following seven parts:**
  - Introduction and explanation of the project
  - Team organization
  - Risk analysis
  - Hardware, software, and human resource requirements
  - Task list and size and effort estimates
  - Project schedule
  - Project monitoring and reporting mechanisms, collectively known as project oversight

An agile project plan

- is feature-based (remember it's built around getting production code running quickly).
- is organized into iterations.
- is multi-layered (because it knows that things will change and that the initial set of requirements are not complete or detailed enough).
- is owned by the team, not the project manager.

Project Organization

- The project organization section of the plan contains the following three things:
  - How you're going to organize the team
  - What process model the project will be using
  - How will the project be run on a day-to-day basis

Risk Analysis

- In the risk analysis section, you need to think about the bad things.

Some risks to watch out for include the following:

- ***Schedule slips*:** That task that you estimated would take three days has just taken three weeks.
- ***Defect rate is excessive*:** Your testing is finding lots of bugs.
- ***Requirements misunderstood*:** What you're doing isn't what the customer wanted.
- ***Requirements churn*:**Requirements churn is probably the largest single reason for missed delivery dates, high defect rates, and project failure.
- ***Turnover*:** Your most experienced developer decides to join a startup three weeks before product delivery.
-

Resource Requirements

- How many people do you need for the project?

Task Estimates

- The first step toward a project schedule is seeing what you'll be doing and figuring out how long each step will take.

Project Schedule

- Once you have estimates of the tasks in your first release or iteration and have people resource estimates, you can create a schedule.
  - Get your developers to tell you the *dependencies* between tasks.
  - Figure out what your *duty cycle* is.
  - Take weekends, vacations, sick days, training, and slack into account when you're making the schedule.
  - You can't schedule a developer to work on two tasks at the same time.

Project Oversight

- Project oversight is what happens once you've got a schedule and your project is underway.

Status Reviews and Presentations

- Status reviews and presentations are an inescapable part of any project. The bigger the project, the
more formal the review.

Defects

- Inevitably, you'll introduce defects (errors) into your program.

- As a developer, your aim is twofold:
  - Introduce as few defects as possible into the code you write.
  - Find as many of them as you can before releasing the code.

Defect levels

- *Fatal*: Either this defect causes the product to crash, or a fundamental piece of functionality doesn't work (for example, you can't save files in your new word processor).
- *Severe*: A major piece of functionality doesn't work, and there's no workaround for it that the user can perform (such as Cut and Paste don't work at all).
- *Serious*: A piece of functionality doesn't work, but there is a workaround for it that the customer can perform (for example, the keyboard shortcuts for Cut and Paste don't work, but the pull-down menus do).
- *Annoying*: There's a minor defect or error in the documentation that may annoy the user, but doesn't affect how the program works (say, Paste is always spelled Patse).
- *New feature request*: This isn't a defect, but a request for the product to do something new (as in, the word processor should have a speech-to-text feature built in).

The Retrospective

- A *retrospective*, as the name implies, is an opportunity to reflect on the project just completed and answer a few questions.

The Retrospective

- Typically, the questions will be like the following:
  - What went right? Did our process work the way we anticipated? Did we meet our schedule? Did we implement all the features required by the customer?
  - What went wrong? Why did we have so many defects? Why did we need to work 60-hour weeks for the last month of the project?
  - What process issues came up? Did we follow our process? If not, what parts were problematic?
  - What do we need to fix for next time? Given the answers to the preceding questions, what do we need to fix in our process, work habits, or environment for the next project?
  - Who's responsible for the fixes? Someone has to be responsible for the changes to our process—who is it? (Don't make it a manager; the development team should own the process).

Summary

- So where do we end up? We've gone through the general parts of managing projects, and I've presented some alternative ways of doing project management. The most important ideas to consider are that *the developers should own the process* and *management should be supportive and listen to the developers*—particularly where schedules and estimates are concerned—and be the buffer between the developers and the world. If you can work in an organization where those things are true, be a happy camper, because you'll be able to write great code.

End