NOTE

Having a good set of automated tests for your system allows you to make fundamental architectural changes with minimal risk. It gives you space in which to work.

Another major benefit of the unit tests was their remarkable shaping of the code design: they practically enforced good structure. Each small code component was crafted as a well-defined entity that could stand alone, as it had to be constructible in a unit test without requiring the rest of the system to be built up around it. Writing unit tests ensured that each module of code was internally cohesive and loosely coupled from the rest of the system. The unit tests forced careful thought about each unit's interface, and ensured that the unit's API was meaningful and internally consistent.

NOTE

Unit testing your code leads to better software designs, so design for testability.

Time for design

One of the contributing factors to Design Town's success was the allotted development timescale, which was neither too long nor too short (just like Goldilocks's porridge). A project needs a conducive environment in which to thrive.

Given too much time, programmers often want to create their magnum opus (the kind of thing that will always be *almost* ready, but never quite materializes). A little pressure is a wonderful thing, and a sense of urgency helps to get things done. However, given too little time, it simply isn't possible to achieve any worthwhile design, and you'll get only a half-baked solution rushed out—just like the Metropolis.

NOTE

Good project planning leads to superior designs. Allot sufficient time to create an architectural masterpiece—they don't appear instantly.

Working with the design

Although the codebase was large, it was coherent and easily understood. New programmers could pick it up and work with it relatively easily. There were no unnecessarily complex interconnections to understand, or weird legacy code to work around.

Since the code has generated relatively few problems and is still enjoyable to work with, there has been very, very low turnover of team members. This is due in part to the developers taking ownership of the design and continually wanting to improve it.

It was interesting to observe how the development team dynamics followed the architecture. Design Town project principles mandated that no one "owned" any area of the design, meaning that any developer could work anywhere in the system. Everyone was expected to write