However, it was built in a very different way, and so the internal structure worked out very differently.

I was involved with the Design Town project from the very start. A brand-new team of capable developers had been assembled to build it from scratch. The team was small (initially four programmers) and, like the Metropolis, the team structure was flat. Fortunately, there was none of the interpersonal rivalry apparent in the Metropolis project, or any vying for positions of power in the team. The members didn't know each other well beforehand and didn't know how well we'd work together, but we were all enthused about the project and relished the challenge.

So far, so good.

Linux and C++ were early decisions for the project, and that shaped the team that had been assembled. From the outset the project had clearly defined goals: a particular first product and a roadmap of future functionality that the codebase had to accommodate. This was to be a general-purpose codebase that would be applied in a number of product configurations.

The development process employed was eXtreme Programming (or XP) (Beck and Andres 2004), which many believe eschews design: *code from the hip, and don't think too far ahead*. In fact, some observers were shocked at our choice and predicted that it would all end in tears, just like the Metropolis. But this is a common misconception. XP does not discourage design; it discourages work that isn't necessary (this is the YAGNI, or *You Aren't Going To Need It*, principle). However, where upfront design is required, XP requires you to do that. It also encourages rapid prototypes (known as *spikes*) to flesh out and prove the validity of designs. Both of these were very useful and contributed greatly to the final software design.

First Steps into Design Town

Early in the design process, we established the main areas of functionality (these included the core audio path, content management, and user control/interface). We considered where they each fit in the system, and an initial architecture was fleshed out, including the core threading models that were necessary to achieve performance requirements.

The relative positions of the separate parts of the system was established in a conventional layer diagram, a simplified part of which is shown in Figure 2-2. Notice that this was *not* a big upfront design. It was an intentionally simple conceptual model of the Design Town: just some blobs on a diagram, a basic system design that could grow easily as pieces of functionality were added. Although basic, this initial architecture proved a solid basis for growth. Whereas the Metropolis had no overall picture and saw functionality grafted (or bodged) in wherever was "convenient," this system had a clear model of what belonged where.

Extra design time was spent on the heart of the system: the audio path. It was essentially an internal subarchitecture of the entire system. To define this, we considered the flow of data through a series of components and arrived at a filter-and-pipeline audio architecture, similar