high-quality code. Whereas the Metropolis was a sprawling mess created by many uncoordinated, fighting programmers, Design Town was clean and cohesive, closely cooperating software components created by closely cooperating colleagues. In many ways, Conway's Law[*] worked in reverse, and the team gelled together as the software did.

NOTE

A team's organization has an inevitable affect on the code it produces. Over time, the architecture also affects how well the team works together. When teams separate, the code interacts clumsily. When they work together, the architecture integrates well.

Where Is It Now?

After some time, the Design Town architecture looked like Figure 2-4. That is, it was remarkably similar to the original design, with a few notable changes—and a lot more experience to prove the design was right. A healthy development process, a smaller, more thoughtful development team, and an appropriate focus on ensuring consistency led to an incredibly simple, clear, and consistent design. This simplicity worked to the advantage of the Design Town, leading to malleable code and rapidly developed products.

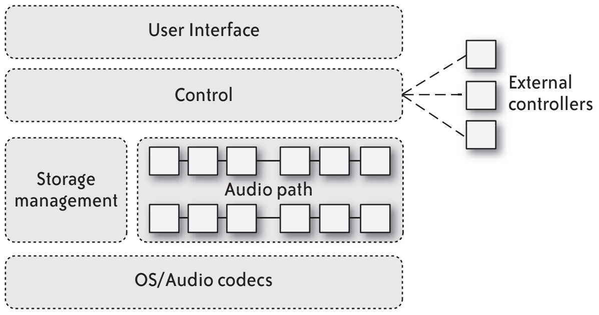

FIGURE 2-4. The Design Town final architecture

At the time of this writing, the Design Town project has been alive for three years. The codebase is still in production use and has spawned a number of successful products. It is still being developed, still growing, still being extended, and still being changed daily. Its design next month might be quite different from how it looks this month, but it probably won't.

Let me make this clear: the code is by no means perfect. It has areas of technical debt that need work, but they stick out against the backdrop of neatness and will be addressed in the future. Nothing is set in stone, and thanks to the adaptable architecture and flexible code structure,

[*] Conway's Law states that code structure follows team structure. Simply stated, it says, "If you have four groups working on a compiler, you'll get a four-pass compiler."