- Code/design reviews for anything not pair-programmed
- Unit tests for every piece of code

These processes ensured that the system never had an incorrect, badly fitting change applied. Anything that didn't mesh with the software design was rejected. This might sound draconian, but they were processes that the developers bought into.

This buy-in highlights an important attitude: the developers believed in the design, and considered it important enough to protect. They took ownership of, and personal responsibility for, the design.

> NOTE
> **Architectural quality must be maintained. This can happen only when the developers are given and take responsibility for it.**

Managing technical debt

Despite these quality control measures, Design Town development was fairly pragmatic. As deadlines approached, a number of corners were cut to allow projects to ship on time. Small code "sins" or design warts were allowed to enter the codebase, either to get functionality working quickly or to avoid high-risk changes near a release.

However, unlike the Messy Metropolis project, these fudges were marked as *technical debt* and scheduled for later revision. These warts stood out clearly, and the developers were not happy about them until they were dealt with. Again, we see the developers taking responsibility for the quality of the design.

Unit tests shape design

One of the core decisions about the codebase (which is also mandated by XP development) was that everything should be unit tested. Unit testing brings many advantages, one of which is the ability to change sections of the software without worrying about destroying everything else in the process. Some areas of the Design Town internal structure received quite radical rework, and the unit tests gave us confidence that the rest of the system had not been broken. For example, the thread model and interconnection interface of the audio pipeline was changed fundamentally. This was a serious design change relatively late in the development of that subsystem, but the rest of the code interfacing with the audio path continued executing perfectly. The unit tests enabled us to change the design.

This kind of "major" design change slowed down as Design Town matured. After an amount of design rework, things settled down, and subsequently there were only minor design changes. The system developed quickly, in an iterative manner, with each step improving the design, until it reached a relatively stable plateau.