The Story Unfolds

Once the initial design had been established by the team, the Design Town project proceeded following the XP process. Design and code construction was either done in pairs or carefully reviewed to ensure that work was correct.

The design and the code developed and matured over time, and as the story of Design Town unfolded, there were the following consequences.

Locating functionality

With a clear overview of the system structure in place from the very beginning, new units of functionality were consistently added to the correct functional areas of the codebase. There was never a question about where code belonged. It was also easy to find the implementation of existing functionality in order to extend it or to fix problems.

Now, sometimes putting new code in the "right" place was harder than simply bodging it into a more convenient, but less tasteful, place. So the existence of an architectural plan sometimes made the developers work harder. The payoff for this extra effort was a *much* easier life later on, when maintaining or extending the system—there was very little cruft to trip over.

> NOTE
> **An architecture helps you to locate functionality: to add it, to modify it, or to fix it. It provides a template for you to slot work into and a map to navigate the system.**

Consistency

The entire system was consistent. Every decision at every level was taken in the context of the whole design. The developers did this intentionally from the outset so all the code produced matched the design fully, and matched all the other code written.

Over the project's history, despite many changes ranging across the entire scope of the codebase—from individual lines of code to the system structure—everything followed the original design template.

> NOTE
> **A clear architectural design leads to a consistent system. All decisions should be made in the context of the architectural design.**

The good taste and elegance of the top-level design naturally fed down to the lower levels. Even at the lowest levels, the code was uniform and neat. A clearly defined software design ensured that there was no duplication, that familiar design patterns were used throughout, familiar interface idioms were adopted, and that there were no unusual object lifetimes or odd resource management issues. Lines of code were written in the context of the town plan.