**NOTE**

**It's important to maintain the quality of a software design. Bad architectural design leads to further bad architectural design.**

### Lack of cohesion

The system's components were not at all cohesive. Where each one should have had a single, well-defined role, instead each component contained a grab bag of functionality that wasn't necessarily related. This made it hard to determine why a component existed at all, and hard to work out where a particular piece of functionality had been implemented in the system.

Naturally, this made bug fixing a nightmare, which seriously affected the quality and reliability of the software.

Both functionality and data were located in the wrong place in the system. Many things you'd consider "core services" were not implemented in the hub of the system, but were simulated by the outlying modules (at great pain and expense).

Further software archaeology showed why: there had been personality struggles in the original team, and so a few key programmers had begun to build their own little software empires. They'd grab the functionality they thought was cool and plonk it into their module, even if it didn't belong there. To deal with this, they would then make ever more baroque communication mechanisms to stitch the control back to the correct place.

**NOTE**

**The health of the working relationships in your development team will feed directly into the software design. Unhealthy relationships and inflated egos lead to unhealthy software.**

### COHESION AND COUPLING

Key qualities of software design are *cohesion* and *coupling*. These are not newfangled "object-oriented" concepts; developers have been talking about them for many years, since the emergence of structured design in the early 1970s. We aim to design systems with components that have:

*Strong cohesion*

Cohesion is a measure of how related functionality is gathered together and how well the parts *inside* a module work as a whole. Cohesion is the glue holding a module together.

Weakly cohesive modules are a sign of bad decomposition. Each module must have a clearly defined role, and not be a grab bag of unrelated functionality.

*Low coupling*

Coupling is a measure of the interdependency *between* modules—the amount of wiring to and from them. In the simplest designs, modules have little coupling and so are less reliant on one