43 It's Always the Goddamned Interfaces

Project team members focus relentlessly on interfaces, both automated and human.

In order to design a system, we must know the interfaces between the system and the environment. We need to know the system's raw inputs and its final products. Until we have that census of inputs and outputs, we are in preliminary analysis: We have not bounded the problem. Once we have that census, we can start to define the functionality of the system.

What happens in design, after we get agreement on functionality? We break a large, complex system into subsystems, and subsystems into components. And yes, a useful way to confine these subsystems and components is to enumerate in turn each of their unique inputs and outputs.

How do we divide the work of implementation? By subsystem and/or by component. A team may take a subsystem to work on, and individuals will build and test components. The subsystem and component boundaries define the edge of the work, the exact responsibility of each developer. These interfaces are contracts between components; one says to another, "You give me exactly this data, only under these conditions, and I will create precisely this product, to be stored in that specific location."

In the early stages of the project, before you have dug into all the minute details, declaring an interface with implementable exactitude can be very difficult, and it can be even more difficult to realize that a nuance has gone unnoticed. Clearly, leaving any interface undefined is no solution at all, so we must define each, to the best of our understanding at the time.

All this adds up to the strong possibility of having interface defects that impact at least two, and probably more, components, and are usually the toughest to deal with.

Teams that know this pattern attack the interfaces early. They build threads of code that exercise interfaces before they have committed to all the components' code. They integrate individuals' code early, and they test often.

> *We met with a single project that had three work groups—one in Canada, one in the U.S., and one in Israel. The manager had on the project intranet a document he called "the Interface Bible." It was the sole document of record of all system interfaces; anything else about any interface was irrelevant. He swore by his bible, so he didn't need to swear about the god-damned interfaces. —TRL*

Managers who know this pattern pay attention to the interfaces of the project team, battling the possibility that any group could be making false assumptions about any interface. Remember Conway's Law: *The product will reflect the organizational structure that produced it.* This is particularly true of the interfaces: Complex human interfaces on the project are liable to result in complex product interfaces.