One Last Compile...

Used and Abused

There comes a time in every project, usually accompanied by a heavy, sick feeling in the stomach, when you realise that, despite your best intentions and extensive pre-planning (ten minutes scribbling on a napkin), your program has turned into a bit of a mess. There are many reasons for this, but in particular I've noticed a tendency for my units to be far too promiscuous in their coupling. I am something of a lout when it comes to my `uses` clause, and Delphi is also cheerfully relaxed in this area (*'Unit X references Unit Y, which is not in your 'uses' clause. Would you like me to add it, even though in six weeks time this will cause you immense pain and frustration?'*).

I have one database program where I fondly imagined that every form I wrote could be reused in other projects which shared the same underlying database. It would be a golden new age of reuse and high productivity. Then I tried to actually reuse some forms, and it would go something like this:

- *Add Forms to New Project.* Usually no problems here, sometimes there would be a slight feeling of smugness, but no other danger signs.
- *Try and open forms.* This often results in strange error message referring to an experimental component that can no longer be found. So I would spend twenty minutes re-writing the code so the form wouldn't need that component, trying not to think about whether the original program would still work next time I tried to compile it. If you start thinking like that it's only a short step to putting in conditional compiler directives, and we all know how dangerous they are, right kids?
- *With form open, and at least giving the appearance of functionality, try and compile.* For the first twenty seconds or so I would nod happily as the compiler blazed on, averting my eyes from the dizzying rise of the hints and warnings count (which can always be remedied at some indeterminate later date). Then, suddenly, I hit the wall. The gods snigger. The compiler is apologetic but firm: *'Cannot find unit foo.dcu'*. This, and it's really quite spooky, never seems to occur in a unit with any reference to unit `foo`. Come to that, a search through all the files in the project does not reveal a single reference to unit `foo`. All that you know is that, somewhere, a unit references a unit which probably references goodness knows how many other units, before finally you come to a unit which references `foo`. (What `foo` does, and why I should care, have long since disappeared in the mists of time.)

All I know is that my project won't compile until I have comprehensively de-`foo`'d my program, or at least told it where to find `foo`. This latter is sometimes what I use to get out of trouble, except even I get uneasy when my library path contains the name of every directory on my drive. So what I do, and if you know a better way to get round this, please write and tell me, is do a Windows search for all files on my hard disk which contain the word '`foo`'. I carefully write down the names of these files. Then I take each filename in turn and do a search for all the files which contain references to them, hoping that one of them might be at least distantly related to my project. Eventually, the path to `foo` is found, and I take another small step towards getting my program to compile. And then I usually find that my executable is 12Mb because it contains every unit I've ever written.

This, I suspect, is not achieving the kinds of high productivity levels that Delphi's designers had in mind when they delivered their baby to the world. No form is an island, and `uses` clauses are so seductive, and so easy, that achieving low coupling in a Delphi project seems next to impossible. A failing on my part? Almost certainly. Can I blame at least some of it on Delphi? Oh, I think so.