PAUL GRAHAM

Why Arc Isn't Especially Object-Oriented

There is a kind of mania for object-oriented programming at the moment, but some of the smartest programmers I know are some of the least excited about it.

My own feeling is that object-oriented programming is a useful technique in some cases, but it isn't something that has to pervade every program you write. You should be able to define new types, but you shouldn't have to express every program as the definition of new types.

I think there are five reasons people like object-oriented programming, and three and a half of them are bad:

1. Object-oriented programming is exciting if you have a statically-typed language without lexical closures or macros. To some degree, it offers a way around these limitations. (See Greenspun's Tenth Rule.)

2. Object-oriented programming is popular in big companies, because it suits the way they write software. At big companies, software tends to be written by large (and frequently changing) teams of mediocre programmers. Object-oriented programming imposes a discipline on these programmers that prevents any one of them from doing too much damage. The price is that the resulting code is bloated with protocols and full of duplication. This is not too high a price for big companies, because their software is probably going to be bloated and full of duplication anyway.

3. Object-oriented programming generates a lot of what looks like work. Back in the days of fanfold, there was a type of programmer who would only put five or ten lines of code on a page, preceded by twenty lines of elaborately formatted comments. Object-oriented programming is like crack for these people: it lets you incorporate all this scaffolding right into your source code. Something that a Lisp hacker might handle by pushing a symbol onto a list becomes a whole file of classes and methods. So it is a good tool if you want to convince yourself, or someone else, that you are doing a lot of work.

4. If a language is itself an object-oriented program, it can be extended by users. Well, maybe. Or maybe you can do even better by offering the sub-concepts of object-oriented programming a la carte. Overloading, for example, is not intrinsically tied to classes. We'll see.

5. Object-oriented abstractions map neatly onto the domains of certain specific kinds of programs, like simulations and

CAD systems.

I personally have never needed object-oriented abstractions. Common Lisp has an enormously powerful object system and I've never used it once. I've done a lot of things (e.g. making hash tables full of closures) that would have required object-oriented techniques to do in wimpier languages, but I have never had to use CLOS.

Maybe I'm just stupid, or have worked on some limited subset of applications. There is a danger in designing a language based on one's own experience of programming. But it seems more dangerous to put stuff in that you've never needed because it's thought to be a good idea.

- Rees Re: OO
- Spanish Translation