SOFTWARE ABSTRACTIONS

Daniel Jackson · February 2, 2005

picking good abstractions

software is built on abstractions

pick good ones, and you get
› clean interfaces (they're simple and fit well)
› maintainable code (they match the problem)
› a more useable system (they're the user's concepts)

pick bad ones, and you get
› a mess that gets worse over time
› the only refactoring that works is starting over
› special cases, hard to use

what makes a good abstraction?

simplicity
> a few small notions, uniformly combined
> expressed clearly & succinctly

I conclude there are two ways of constructing a software design: One way is to make it so simple there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies.

-- Tony Hoare, Turing Award Lecture, 1980

what makes a good abstraction?

depth
> captures the complexities that matter

To design something really well, you have to get it. You have to really grok what it's all about. It takes a passionate commitment to really thoroughly understand something, chew it up, not just quickly swallow it. Most people don't take the time to do that.

-- Steve Jobs, The Wired Interview, Issue 4.02, Feb 1996

approach

a modelling notation
› Alloy, based on relational logic
› simple and small but expressive

an analysis
› fully automatic
› covers huge space
› user specifies properties, not test cases

patterns
› of modelling and analysis

what's wrong with just sketching?

wishful thinking
› biggest risk in designing abstractions
› to overcome, need precision & analysis

The first principle is that you must not fool yourself and you are the easiest person to fool.
-- Richard P. Feynman

what's wrong with just coding?

nothing, so long as

- you're prepared to start again
- you like reading other people's code
- you like writing test cases
-

Another strength of design with pictures is speed. In the time it would take you to code one design, you can compare and contrast three designs using pictures. The trouble with pictures, however, is that they can't give you concrete feedback. The XP strategy is that anyone can design with pictures all they want, but as soon as a question is raised that can be answered with code, the designers must turn to code for the answer. The pictures aren't saved.

-- Kent Beck, in *Extreme Programming Explained*