Composite Languages

Andrew Hirsch

The George Washington University

Thursday, January 17, 2013

Business issues

- Two businesses want to make a deal
- What needs to happen?
  - Negotiate a deal
  - Sign the paperwork
- What can be bad?
  - Misunderstandings can be devastating
  - Nothing should be binding until the paperwork is filled

Communication is key

- How do businesses make sure nothings wrong?

Communication is key

- How do businesses make sure nothings wrong?

- They use lawyers
- Lawyers know how to speak to each other
  - Makes bad things happen less often

Communication among thieves (well, lawyers)

- Lawyers need to make misunderstandings not happen
- Use appropriate protocols
- Now, everyone knows what that means!
- Companies do not talk without going through lawyers

Now you've gone and put your foot in it

- What happens when things DO go wrong?

Now you've gone and put your foot in it

- What happens when things DO go wrong?
- In come the lawyers again
- The lawyers can come in and fix the mistake
- Hopefully, the other businesses don't even notice (These lawyers aren't the most moral sort)

What on earth has this got to do with computers?

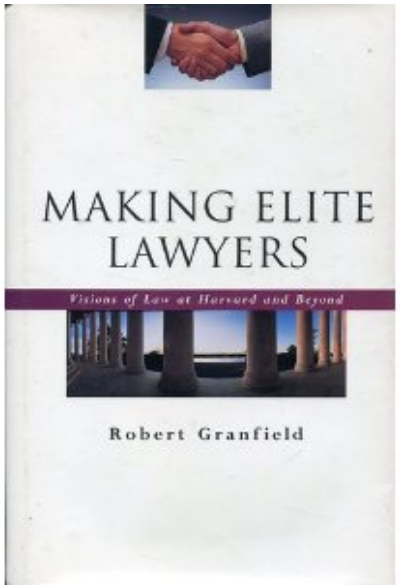
- Now you know my opinions on lawyers
- Let's get down to senior design
- Working in the `Composite` operating system
- `Composite` has different independent parts, *components*
- These are like the businesses from that last example.

So that was just bias against lawyers?

- Where do the lawyers come in to this?
- The components need to be able to talk with each other
- Things shouldn't go wrong
- If they do, they should go right again
- No other components should notice!

So about that project...

- I'm building the lawyers



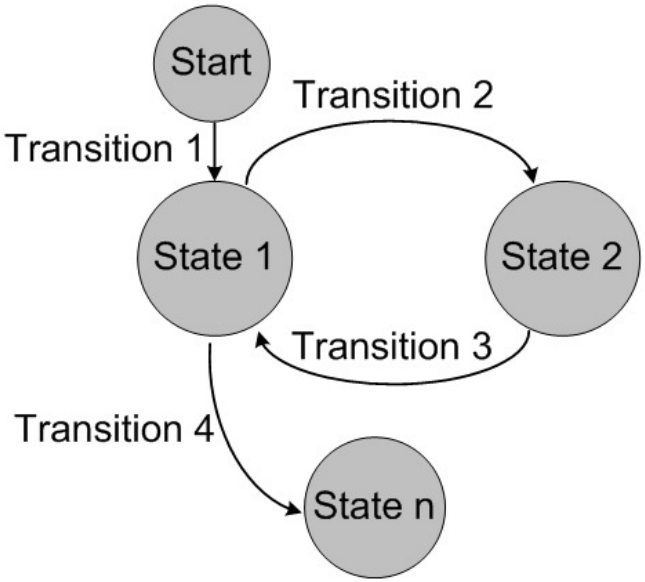
- Lawyers = *stubs*: Code that allows components to talk
- Before: stubs are generated by hand
- Now: generated automatically

Automatic lawyer machine

- In composite, every component implements an *interface*
- This specifies what they can do as a list of functions
- Every function has a stub generated for it
- This project: language for describing interfaces:
  IDL: Interface Description Language
  (Such a creative name, I know)

How do lawyers talk to each other?

- Every interface also describes a *protocol*



- Protocols can be thought of as how components communicate
- Similar to the protocols lawyers use
- These are also described in the IDL
  (So it's not so uncreatively named after all!)

Cleaning up the messes

- What happens when a component fails?
- It should "un-fail"
- Other components shouldn't notice!
- Understand what others expect it to
- Back to the same spot in IDL protocols
- Code generated automatically!

Demonstration