70% Demonstration

Andrew Hirsch

The George Washington University

Thursday, January 17, 2013

The Life of a spy

- Imagine a spy, Alice
- She has to pass messages on to her fellow, Bob
- But they can only be so long!
- She can tell the spy where to go for a (longer) message
- She packages up the longer messages and tells Bob where to go

The troubles of space

- But, now, Alice has a limited area she is permitted to move in (her handlers keep her on a tight leash)
- She only have a bit of overlap with Bob
- She has to give them directions to get to a longer message in the shared area

The Problem with Composite Functions

- This is exactly what happens in the Composite operating system
- Composite is made up of "components", which only have a limited shared space
- They can only talk to each other in such long messages

Coding the Solution

- The solution is worse than the problem
- There is a lot of sticky code that has to be perfectly hand-written!
- This is where we automate

Automating stub generation

- Pony is good at this sort of automation
- Anywhere there's a function that should be called between components, automatically generate *stubs*
- These stubs tell how to package and address messages in shared memory

Is there a need for extra syntax?

- It is common to add extra syntax here
- This is called an *Interface Description Language*
- If we need extra syntax, we will have to add it in to Pony by hand

Protocols

- Let us return to Alice and Bob
- How are they to speak to each other?
- They use a well defined *protocol* that they can each react to
- Then they always know what the other is saying

Composing Machines

- Imagine that there is a machine that given a red widget should always return a blue one
- And a second machine that given a blue widget, should always return a green one
- We can make a green widget from a red one by giving the red widget to the first machine
- Then putting the resulting blue widget in the second!

Contracts

- The kind of promises given by the first machine are known as contracts
- They can be enforced by a programming language when given by a function
- This is important when promises must be kept, such as in an OS

Protocols in Composite

- Composite components should use protocols
- That way they talk to each other in sane ways
- But, they should know how they respond to different inputs
- This should be enforced by contracts

Contract Language

- Functions that are called by other components should implement protocols
- It needs to be enforced that sticks to a protocol
- A protocol language that it is easy to enforce speaking in

The Plan Going Forward

- Create transformation for stub generation
- Design extra syntax (with Composite team)
- Create fork of pony parser to deal with extra syntax
- Create transformations for extra syntax
- Design protocol language (with Composite team)
- Create fork of pony parser to deal with protocol language
- Create transformation for protocol language