A Little Reflection On Metamodels

Jonathan Worthington
Netherlands Perl Workshop 2010
A Little Reflection On Metamodels

OH HAI!
A Little Reflection On Metamodels

Metamodels sound a little scary... 😞
...but don’t worry, I’m just going to tell you a story. 😊
Chapter 1

The anthropomorphic class
Once upon a time, I wrote a class.

class Stroopwafel is Cake {
    has $!area;
    has $.filling;
    method eat() {
        for 1..$area {
            say "om nom nom nom nom nom nom";
        }
    }
}
I thought my work was done, and I could go for a beer.
But then my class started asking me questions...
class Stroopwafel is Cake {
  has !$area;
  has $.filling;
  method eat() {
    for 1..$area {
      say "om nom nom nom nom nom nom";
    }
  }
}

How was I created?
What does it mean to have methods?

class Stroopwafel is Cake {
  has !$area;
  has $.filling;
  method eat() {
    for 1..$area {
      say "om nom nom nom nom nom nom";
    }
  }
}
What does it mean to inherit?

class Stroopwafel is Cake {
    has $!area;
    has $.filling;
    method eat() {
        for 1..$area {
            say "om nom nom nom nom nom nom";
        }
    }
}
class Stroopwafel is Cake {
    has $!area;
    has $.filling;
    method eat() {
        for 1..$area {
            say "om nom nom nom nom nom nom";
        }
    }
}
What about prototype OO?

class Stroopwafel is Cake {
    has (!$area;
    has $.filling;
    method eat() {
        for 1..$area {
            say "om nom nom nom nom nom nom";
        }
    }
}
But I didn’t know how to answer.
Chapter 2

jnthon tries to implement Perl 6 OO
Rakudo development is generally breadth-first.
A Little Reflection On Metamodels

<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
</table>

# A Little Reflection On Metamodels

<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>So Crap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not so bad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Awesomeness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hey nice!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Better than</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stroopwafels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## A Little Reflection On Metamodels

<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>So Crap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not so bad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation Awesomeness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hey nice!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Better than stroopwafels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Before we called it Rakudo
## A Little Reflection On Metamodels

<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>So Crap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not so bad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation Awesomeness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hey nice!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Better than stroopwafels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Hack**
<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>So Crap</td>
<td>📝</td>
<td>📝</td>
<td>📝</td>
</tr>
<tr>
<td>Meh</td>
<td>📝</td>
<td>📝</td>
<td>📝</td>
</tr>
<tr>
<td>Not so bad</td>
<td>📝</td>
<td>📝</td>
<td>📝</td>
</tr>
<tr>
<td>Implementation Awesomeness</td>
<td>📝</td>
<td>📝</td>
<td>📝</td>
</tr>
<tr>
<td>Hey nice!</td>
<td>📝</td>
<td>📝</td>
<td>📝</td>
</tr>
<tr>
<td>Better than stroopwafels</td>
<td>📝</td>
<td>📝</td>
<td>📝</td>
</tr>
</tbody>
</table>

**Hack hack**
## A Little Reflection On Metamodels

<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>So Crap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not so bad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td>Hack hack hack</td>
</tr>
<tr>
<td>Awesomeness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hey nice!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Better than</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stroopwafels</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>OO</td>
<td>Regexes</td>
<td>Built-ins</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>So Crap</td>
<td>🕳️</td>
<td>🕳️</td>
<td>🕳️</td>
</tr>
<tr>
<td>Meh</td>
<td>🕳️</td>
<td>🕳️</td>
<td>🕳️</td>
</tr>
<tr>
<td>Not so bad</td>
<td>🕳️</td>
<td>🕳️</td>
<td>🕳️</td>
</tr>
<tr>
<td>Implementation Awesomeness</td>
<td>🕳️</td>
<td>🕳️</td>
<td>🕳️</td>
</tr>
<tr>
<td>Hey nice!</td>
<td>🕳️</td>
<td>🕳️</td>
<td>🕳️</td>
</tr>
<tr>
<td>Better than stroopwafels</td>
<td>🕳️</td>
<td>🕳️</td>
<td>🕳️</td>
</tr>
</tbody>
</table>

OMG new grammar engine!
## A Little Reflection On Metamodels

<table>
<thead>
<tr>
<th>Feature</th>
<th>OO</th>
<th>Regexes</th>
<th>Built-ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>So Crap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not so bad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation Awesomeness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hey nice!</td>
<td></td>
<td></td>
<td>Mmm....beer and hacking!</td>
</tr>
<tr>
<td>Better than stroopwafels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Means you get a compiler with some coverage of many features...
...meaning that people can start to write programs...
...and then complete and improve features over time.
A Little Reflection On Metamodels

STD.pm

... token package_declarator:class {  
  :my $*PKGDECL := 'class';  
  <sym> <package_def> 
} 

token package_declarator:grammar {  
  :my $*PKGDECL := 'grammar';  
  <sym> <package_def> 
} 

token package_declarator:role {  
  :my $*PKGDECL := 'role';  
  <sym> <package_def> 
} 

...
A Little Reflection On Metamodels

...
First cut(s): needed something that works, so fairly hard coded.
A Little Reflection On Metamodels
A Little Reflection On Metamodels

But...

Having the details all hard-coded bloats the compiler
A Little Reflection On Metamodels

But...
Not extensible, so no way to add more package types in future.
Chapter 3

Metamodels to the rescue!
A Little Reflection On Metamodels

Just when I thought I’d never work it out...
A Little Reflection On Metamodels

...along came Metamodel Man!

OH HAI!
He gave me knowledge of the wonder of metamodels.
So what is a metamodel?
Can understand a word from its parts.
Politics
Politics

Politics
A Little Reflection On Metamodels

Politics

Poli          tics

Latin

Many

Blood sucking creatures
A Little Reflection On Metamodels

Metamodel
A Little Reflection On Metamodels

Metamodel

Meta model
A Little Reflection On Metamodels

Metamodel

Meta model

Because I said so
A Little Reflection On Metamodels

Metamodel

Meta model

Because I said so

Things that describe...

...objects in our system.
Each package type maps to some “meta-package” type

class => Class HOW

role => Role HOW
Compile a class definition...

class Stroopwafel is Cake {
    has !$area;
    has $.filling;
    method eat() {
        for 1..$area {
            say "om nom nom nom nom nom nom";
        }
    }
}
A Little Reflection On Metamodels

...to calls on a meta-class instance.

my $temp = ClassHOW.new('Stroopwafel');
trait_mod:<is>{$temp, Cake};
$temp.^add_attribute(Attribute.new(
   name => '$!filling', has_accessor => True
));
$temp.^add_attribute(Attribute.new(
   name => '$!area`
));
$temp.^add_method('eat', method () {
   ...
});
my $type-object = $temp.^compose();
The semantics of a class are whatever the ClassHOW metaclass decides that they are.
Differences between package types can be encapsulated in the meta-packages.
Means that programmers are able to create their own types of package cleanly.
A Little Reflection On Metamodels

...and they all lived hackily ever after.
A Little Reflection On Metamodels

The End