Cohesion Metrics

Designing and Maintaining Software (DAMS)

Louis Rose

Recap: Cohesion

“A measure of the degree to which the elements of a module belong together.”

- Yourdon & Constantine
Structured Design, 1979

Cohesion Metrics

Indicate the extent to which
elements are used together
(and hence belong together?)

Corollary

There are lots of cohesion metrics!

LCOM

Lack of Cohesion of Methods

An inverse metric: a lower score is better

We are using the 4th revision: $LCOM_4$

Partitions an element (e.g., class) into sets of related elements

- Briand, Daly and Wust

A Unified Framework for Cohesion Measurement in Object-Oriented Systems
Empirical Software Engineering, 3, 1998

$LCOM_4$ Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```

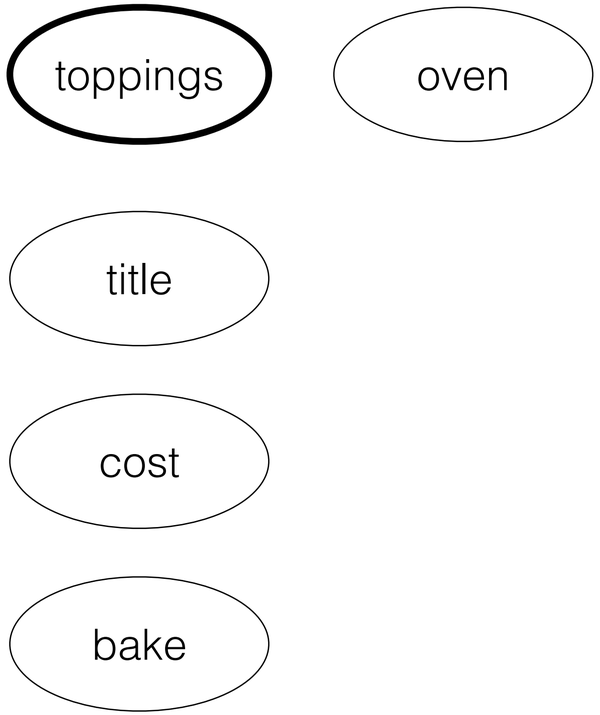
$LCOM_4$ Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



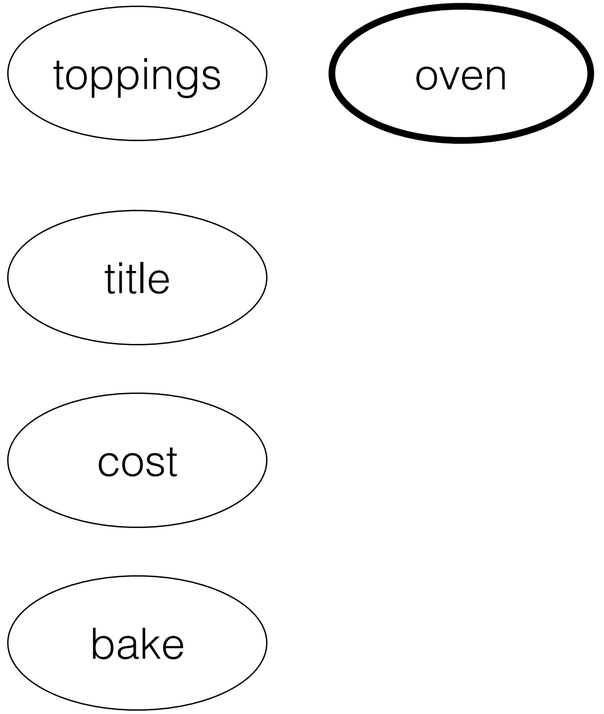
$LCOM_4$ Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



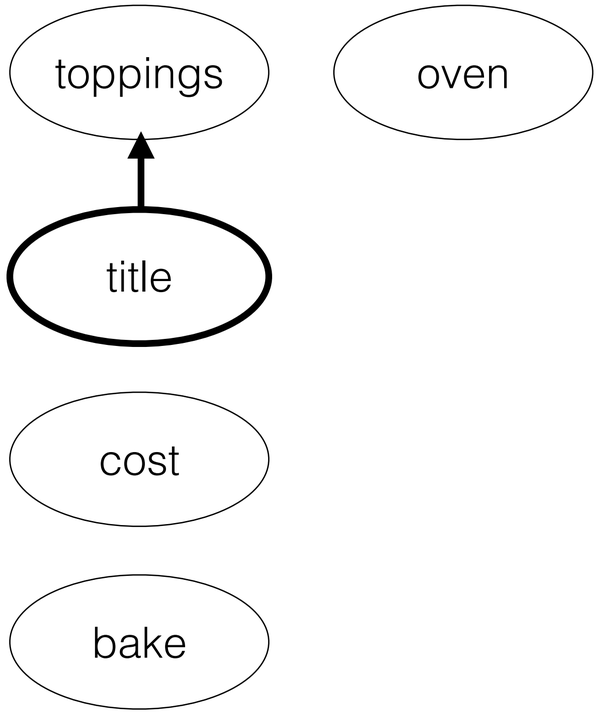
$LCOM_4$ Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



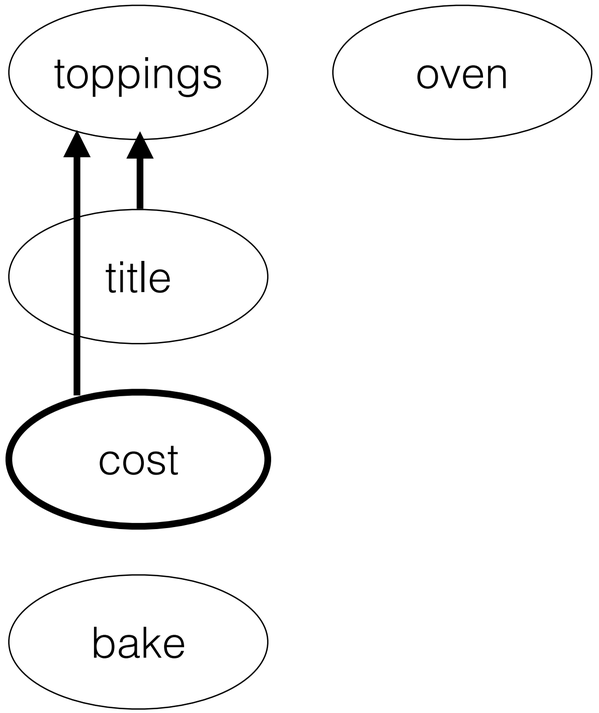
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



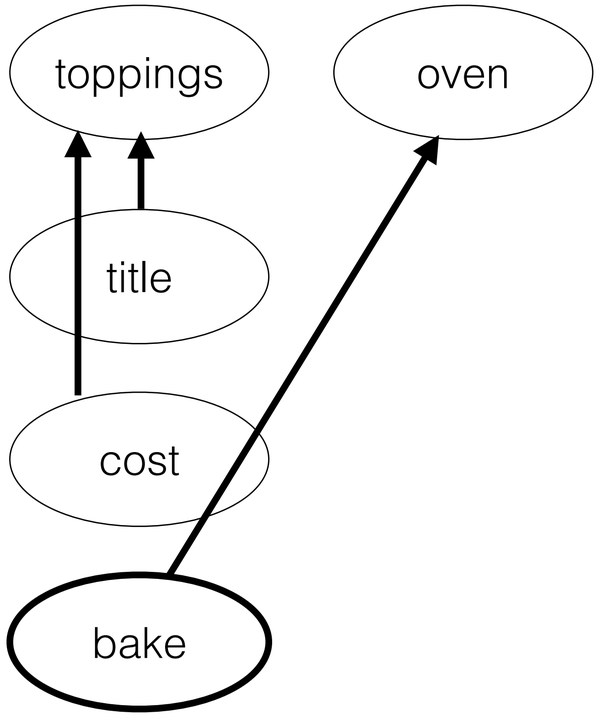
$LCOM_4$ Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



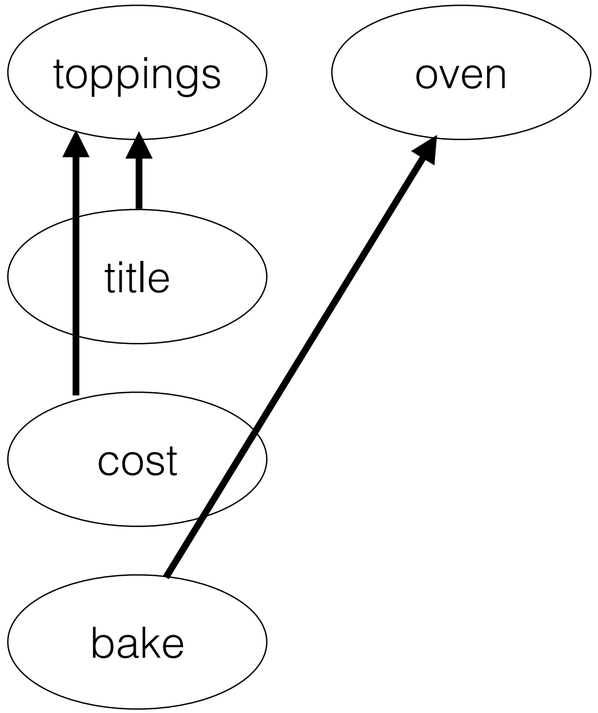
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



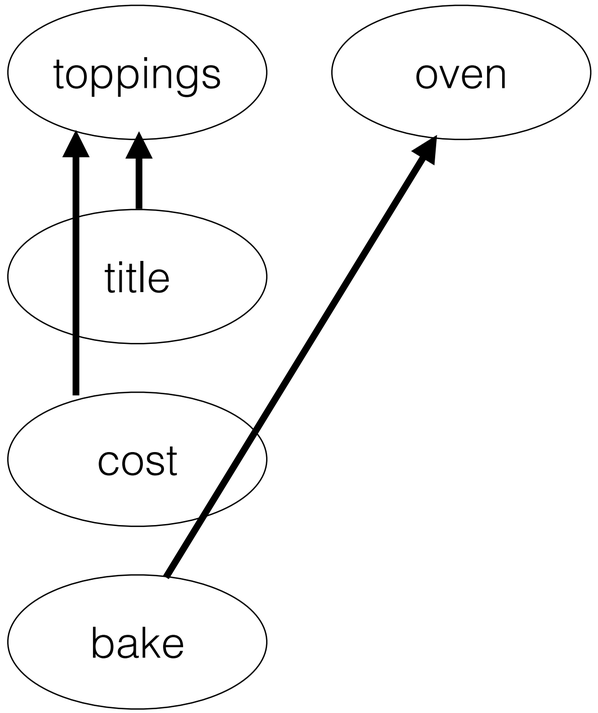
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



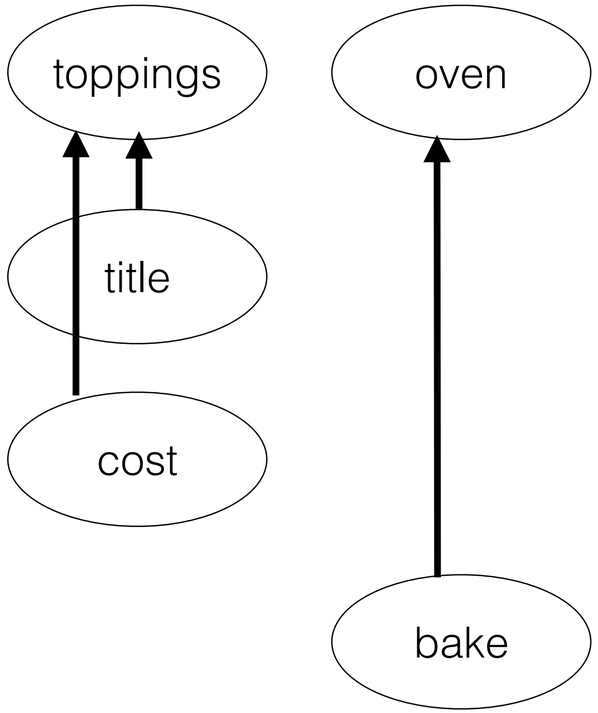
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



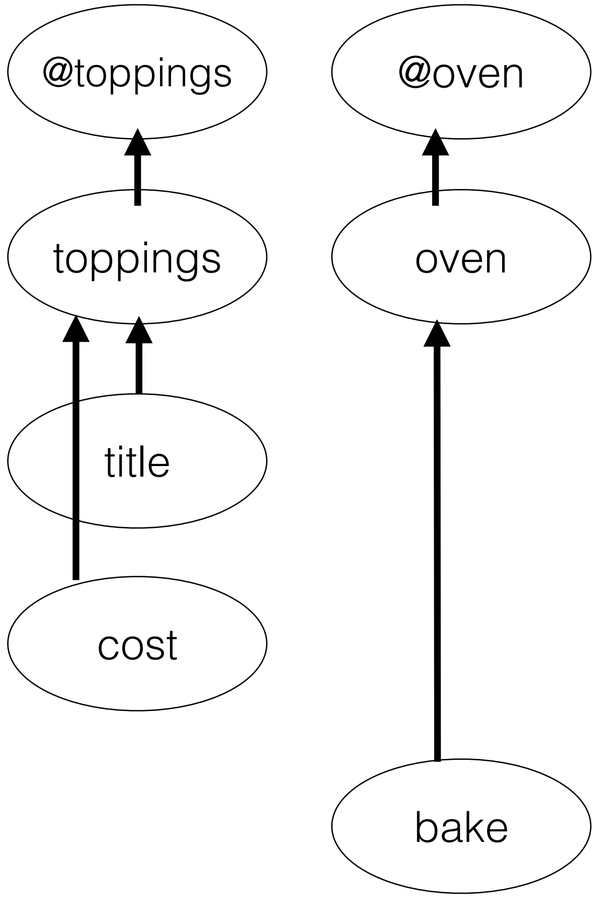
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    oven.bake(self)
  end
end
```



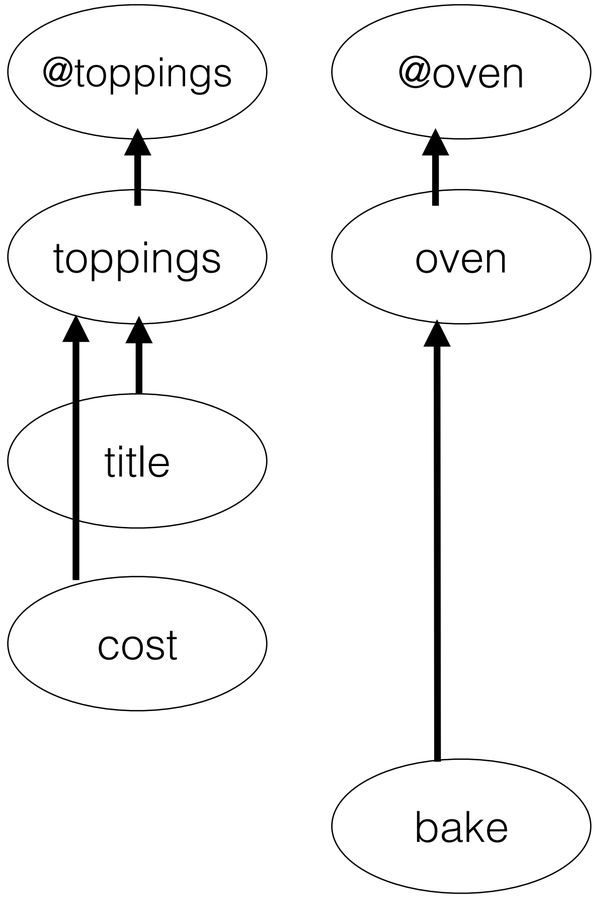
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    @oven.bake(self)
  end
end
```



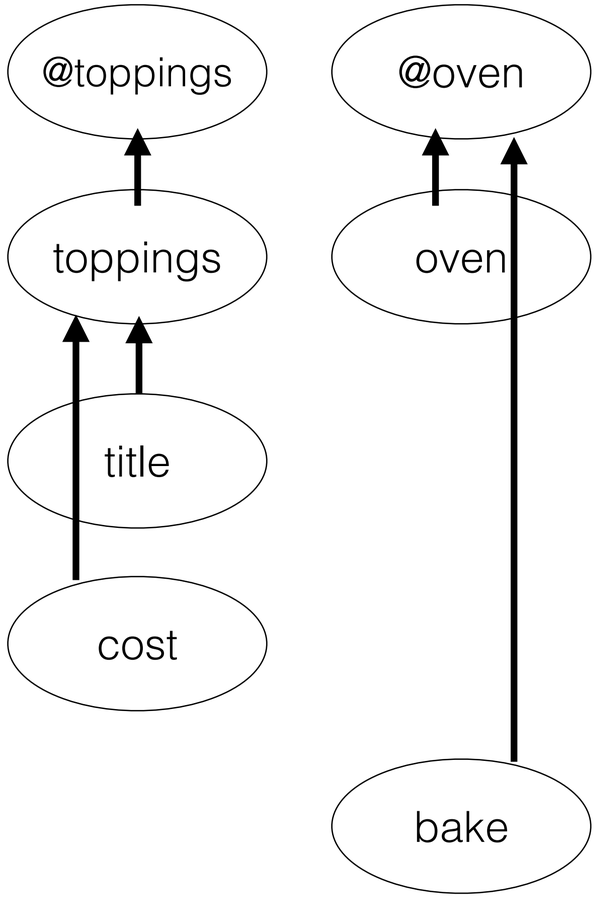
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    @oven.bake(self)
  end
end
```



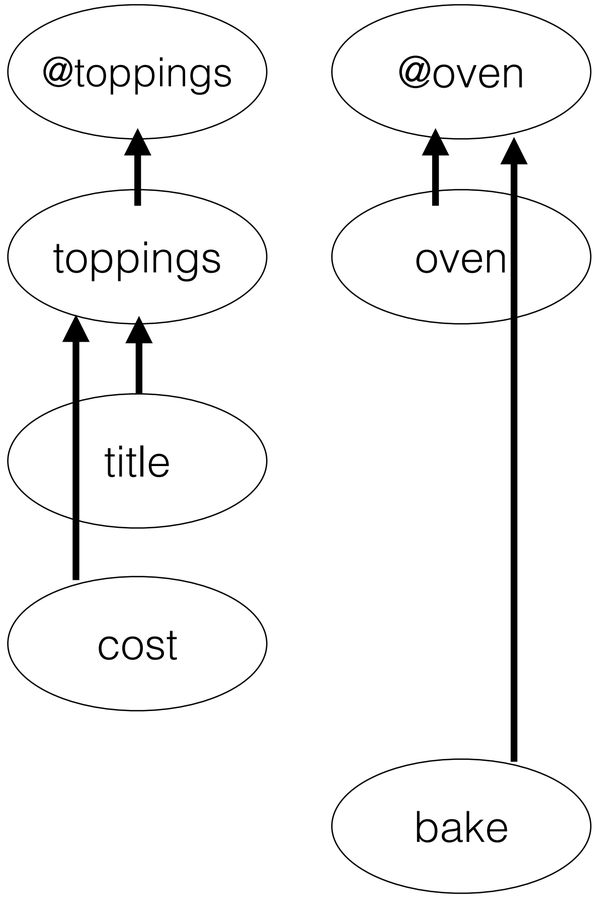
LCOM4 Example

```ruby
class Pizza
  attr_reader :toppings, :oven

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def bake
    @oven.bake(self)
  end
end
```



Divergent Change

What is it?
A class can change for several reasons

Why is it problematic?
Changes are harder to make with confidence

When does it arise?
👎🏼 Violation of Single Responsibility Principle
👍🏼 You know that the triggering changes won’t occur

Data Clumps

What is it?
Several pieces of data are often used together

Why is it problematic?
Behaviour that operates on the clump has no home
(and consequently is often duplicated)

When does it arise?
👎🏼 High cohesion of the clump has not been detected
👍🏼 Domain is not thoroughly understood (yet)

Primitive Obsession

What is it?

Using built-in types to represent domain concepts.

Why is it problematic?

Behaviour that operates on the domain object has no home (and consequently is often duplicated)

When does it arise?

👎🏼 Fear of OO (often due to performance concerns)

👍🏼 If any extracted object would be very small

Feature Envy

What is it?
An object is more concerned with another object's attributes or methods than its own.

Why is it problematic?
Behaviour probably belongs on the envied object (and consequently is often duplicated)

When does it arise?
👎🏼 Violation of Tell, Don't Ask principle
👍🏼 Behaviour is much more likely to change when the envying class changes

Summary

Classes should have a single responsibility:
a single reason to change

Classes with low cohesion normally have
more than one responsibility

Extract classes to redistribute responsibilities