Coupling metrics

Designing and Maintaining Software (DAMS)

Louis Rose

Recap: Coupling

“The measure of the strength of association established by a connection from one module to another.”

- Stevens, Myers and Constantine
Structured Design
IBM Systems Journal 13:2, 1974

Recap: Coupling

*"The measure of the **strength of association** established by a connection from one module to another."*

- Stevens, Myers and Constantine
Structured Design
IBM Systems Journal 13:2, 1974

Strength of Association

There's no consensus on what this means :-(

Type (method, attribute, method types, …) ?
Strength (binary, sum, weighted, …) ?
Directed or undirected?
Direct or indirect?
Stability of dependee?
Includes inheritance?
Classes or objects?

- Briand, Daly, Wust
A unified framework for coupling measurement
in object-oriented systems
IEEE Trans. Software Engineering, 25:1, 1999

Corollary

There are lots of coupling metrics!

Message Passing Coupling

Counts number of “send statements” out from a class

Only counts invocations of methods on other classes

Only counts invocations from “the interface increment”
(i.e., not from inherited methods)

Not clear whether or not to count overridden methods or
whether or not to count invocations of inherited methods

- Li and Henry
Object-oriented metrics that predict maintainability
Object Oriented Programming, 8:4, 1993

Problem

mpc(Pizza) == mpc(Restaurant)?

```
class Pizza
  def title
    @toppings.title
  end

  def cost
    @toppings.cost + 4
  end

   def weight
    @toppings.size * 5 + 2
  end
end
```

```
class Restaurant
  def seat(party)
    seats = party.size
    table = @tables.find_free(seats: seats)
    @waiter.seat(party, table)
  end
end
```

Coupling Between Objects

Counts the number of other classes to which
a class is coupled (other than via inheritance)

$$CBO(c) = |d \in C - (\{c\} \cup Ancestors(C))|\ uses(c,d) \vee uses(d,c)$$

- Chidamber and Kemerer
A metrics suite for object-oriented design
IEEE Transactions on Software Engineering 20:6, 1991

Coupling Between Objects

Counts the number of other classes to which
a class is coupled (other than via inheritance)

$$CBO(c) = |d \in C - (\{c\} \cup Ancestors(C))| \; uses(c,d) \vee uses(d,c)$$

- Chidamber and Kemerer
A metrics suite for object-oriented design
IEEE Transactions on Software Engineering 20:6, 1991

Coupling Between Objects

Counts the number of other classes to which
a class is coupled (other than via inheritance)

$$CBO(c) = |d \in C - (\{c\} \cup Ancestors(C))| \; uses(c,d) \vee uses(d,c)$$

- Chidamber and Kemerer
A metrics suite for object-oriented design
IEEE Transactions on Software Engineering 20:6, 1991

Problem

What type is `@toppings`?

```
class Pizza
  def title
    @toppings.title
  end

  def cost
    @toppings.cost + 4
  end
end
```

```
class Plain
  def title
    "Margherita"
  end
end

class Topped
  def initialize(toppings = [])
    @toppings = toppings
  end

  def title
    @toppings.join(" and ")
  end
end
```

A solution?

What type is `@toppings`?

```
class Pizza
  def title
    @toppings.title
  end

  def cost
    @toppings.cost + 4
  end
end
```

```
class Plain
  def title
    "Margherita"
  end
end

class Topped
  def initialize(toppings = [])
    @toppings = toppings
  end

  def title
    @toppings.join(" and ")
  end
end
```

A solution?

What type is `@toppings`?

```
class Pizza
  def title
    @toppings.title
  end

  def cost
    @toppings.cost + 4
  end
end
```

```
class Plain
  def title
    "Margherita"
  end
end

class Topped
  def initialize(toppings = [])
    @toppings = toppings
  end

  def title
    @toppings.join(" and ")
  end
end
```

Shotgun Surgery

What is it?

Lots of the codebase is affected by a single change

Why is it problematic?

Cost of change is high. Inconsistency is likely to arise.

When does it arise?

👎🏼 Violation of Don’t Repeat Yourself principle

👍🏼 You know that the triggering change won’t occur

Method Chains

What is it?

A long chain of send expressions: `foo.bar.baz.qux`

Why is it problematic?

Couples `self` to `foo`, `bar`, `baz` and `qux`

When does it arise?

👎🏼 Wanting to alter distant state

👍🏼 Querying a domain model

`customer.receipt.pizza.price`

👍🏼 Using built-in libraries:

`foo.select{ … }.collect{ … }.sort`

Law of Demeter

"Talk only to your neighbours"

allowance	example
yourself	`self`
your own toys	`formatter`
toys you've been given	`printable`
toys you've made yourself	`out`

```
class Printer
  attr_reader :formatter

  def print(printable)
    out = TempFile.new
    ...
  end
end
```

Summary

Many different factors can be considered
when designing a coupling metric

MPC can be computed statically, but doesn't
distinguish between tight or diffuse coupling

CBO can, but must be computed at runtime
(in Ruby) and hence cannot guarantee completeness