Object Dimensions

Yegor Bugayenko

Lecture #10 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

Object Metrics

- Number of attributes, methods, constructors, destructors
- Number of static attributes, methods
- Number of types/interfaces
- Number of parent and child classes
- Number of annotations
- Number of static blocks
- Number of nested classes
- Number of type parameters (generics)

Fernando Brito e Abreu

"**Metrics for Object-Oriented Design (MOOD)**: Being able to predict some software quality characteristics based on the design, is one of our great motivations. This ability will allow the designing process to be guided, for instance, by means of heuristics."

— Fernando Brito and Rogério dos Santos Carapuça. Object-Oriented Software Engineering: Measuring and Controlling the Development Process. In *Proceedings of the 4th International Conference on Software Quality*, pages 1–8, 1994

Method Hiding Factor (MHF)

```
class Book {
  public String title() {
    return this.read() + ".";
  }
  private String read() {
    // read from database
  }
}
```

$\mathtt{MHF}_{\mathtt{Book}} = 1/2$

“MHF is defined as the ratio of the sum of the $\underline{\text{invisibilities}}$ of all methods defined in all classes to the total number of methods defined in the system under consideration.”

Attribute Hiding Factor (AHF)

```
class Book {
  public int id;
  private File path;
  public String title() {
    var txt = Files.read(path);
    // 1. Find title by 'id'
    // 2. Return it
  }
}

var b = new Book();
b.id = 42;
var t = b.title();
```

$\texttt{AHF}_{\texttt{Book}} = 1/2$

“AHF is defined as the ratio of the sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the system under consideration.”

Method Inheritance Factor (MIF)

```
class Book extends Material {
  @Override
  public String content() {
    return "Hello, world!";
  }
  public String title() {
    return "David West";
  }
}
```

$\text{MIF}_{\text{Book}} = 1/2$

"MIF is defined as the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods (locally defined plus inherited) for all classes."

Attribute Inheritance Factor (AIF)

```java
class Material {
  protected String content;
}

class Book extends Material {
  private String title;
}
```

$\texttt{AIF}_{\texttt{Book}} = 1/2$

"AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes (locally defined plus inherited) for all classes."

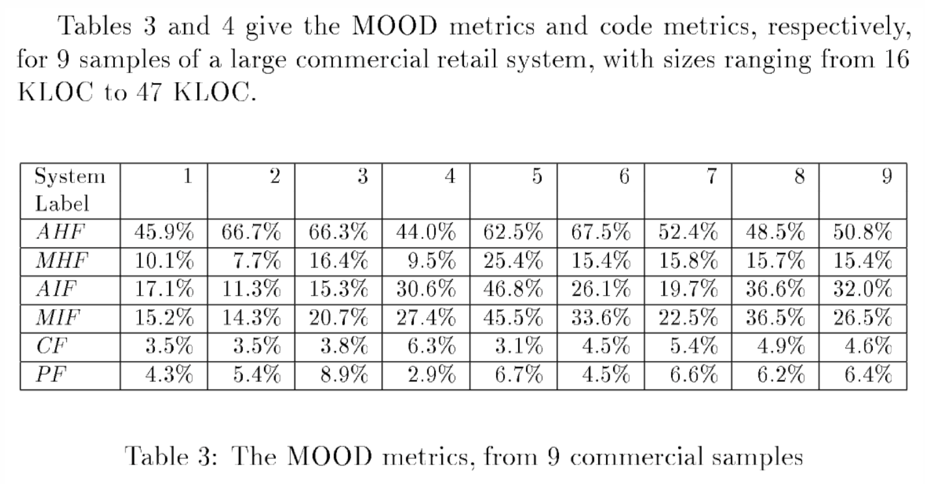
Polymorphism Factor (PF)

```java
class Material {
  public String content() {
    return "Hello, world!";
  }
}

class Book extends Material {
  public String title() {
    return "David";
  }
}
```

$PF_{\text{Book}} = 0/1$

"PF is the number of methods that redefine inherited methods, divided by the maximum number of possible distinct polymorphic situations."



Tables 3 and 4 give the MOOD metrics and code metrics, respectively, for 9 samples of a large commercial retail system, with sizes ranging from 16 KLOC to 47 KLOC.

System Label	1	2	3	4	5	6	7	8	9
AHF	45.9%	66.7%	66.3%	44.0%	62.5%	67.5%	52.4%	48.5%	50.8%
MHF	10.1%	7.7%	16.4%	9.5%	25.4%	15.4%	15.8%	15.7%	15.4%
AIF	17.1%	11.3%	15.3%	30.6%	46.8%	26.1%	19.7%	36.6%	32.0%
MIF	15.2%	14.3%	20.7%	27.4%	45.5%	33.6%	22.5%	36.5%	26.5%
CF	3.5%	3.5%	3.8%	6.3%	3.1%	4.5%	5.4%	4.9%	4.6%
PF	4.3%	5.4%	8.9%	2.9%	6.7%	4.5%	6.6%	6.2%	6.4%

Table 3: The MOOD metrics, from 9 commercial samples

"Results show that the metrics could be used to provide an overall assessment of a software system, which may be helpful to managers of software development projects. However, further empirical studies are needed before these results can be generalised."

Source: Rachel Harrison, Steve J. Counsell, and Reuben V. Nithi. An Evaluation of the MOOD Set of Object-Oriented Software Metrics. *IEEE Transactions on Software Engineering*, 24(6):491–496, 1998. doi:10.1109/32.689404

Chris F. Kemerer

"The inheritance hierarchy, a directed acyclic graph will be described as a tree structure with classes as nodes, leaves and a root. In any design application, there can be many possible hierarchies of classes. Design choices on the hierarchy employed to represent the application are essentially choices about restricting or expanding the scope of properties of the objects in the application."

— Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994. doi:10.1109/32.295895

Weighted Methods per Class (WMC)

Consider a class C_i, with methods $M_1, \ldots, M_n$. Let $c_1, \ldots, c_n$ be the complexity of the methods. Then:

$$\text{WMC} = \sum_{i=1}^{n} c_i$$

If all method complexities are considered to be unity, then $\text{WMC} = n$, the number of methods.

"It can be argued that developers approach the task of writing a method as they would a traditional program, and therefore some traditional static complexity metric, such as McCabe's Cyclomatic number, may be appropriate."

Source: Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994. doi:10.1109/32.295895

Depth of Inheritance Tree (DIT)

```java
class Material {
  public String content() {
    return "Hello, world!";
  }
}

class Book extends Material {
  public String title() {
    return "David";
  }
}
```

$\text{DIT}_{\texttt{Material}} = 0$

$\text{DIT}_{\texttt{Book}} = 1$

"DIT is a measure of how many ancestor classes can potentially affect this class."

Number of Children (NOC)

```java
class Material {
  public String content() {
    return "Hello, world!";
  }
}

class Book extends Material {
  public String title() {
    return "David";
  }
}
```

$\mathtt{NOC}_{\mathtt{Material}} = 1$

$\mathtt{NOC}_{\mathtt{Book}} = 0$

"NOC is a measure of how many sub-classes are going to inherit the methods of the parent class."

Response For a Class (RFC)

```
class Trim
  String value()

class Book
  public String title()
    return new Trim(
      this.read()
    ).value();
  private String read()
    // read from the DB
```

$\mathtt{RFC}_{\mathtt{Book}} = 3$

"RFC is the count of the set of all methods that can be invoked in response to a message to an object of the class or by some method in the class."

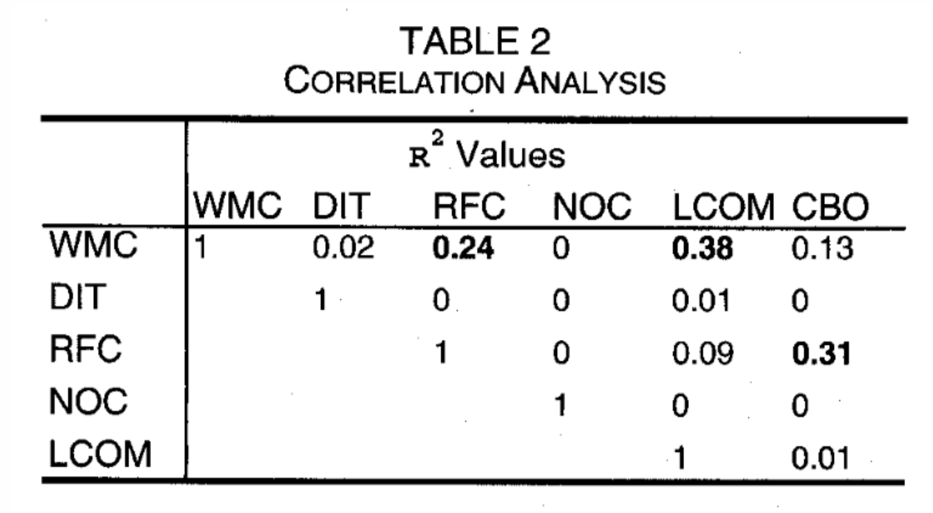
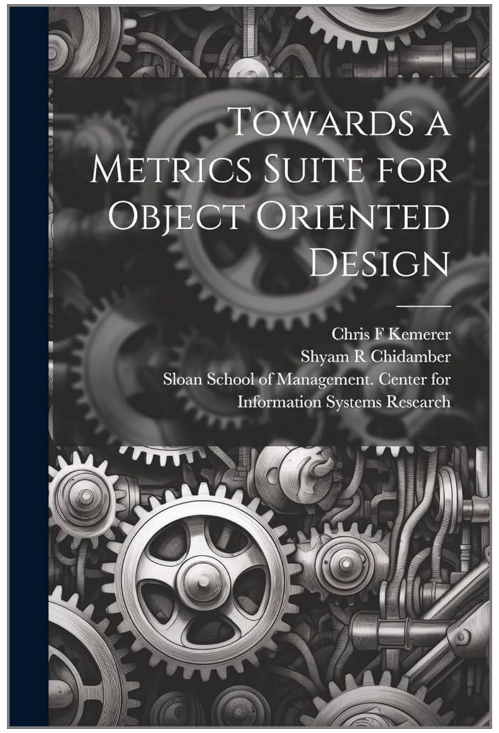


TABLE 2
CORRELATION ANALYSIS

	R^2 Values					
	WMC	DIT	RFC	NOC	LCOM	CBO
WMC	1	0.02	**0.24**	0	**0.38**	0.13
DIT		1	0	0	0.01	0
RFC			1	0	0.09	**0.31**
NOC				1	0	0
LCOM					1	0.01

"Table shows very clearly that linear Pearson's correlations between the studied OO metrics are, in general, very weak. We conclude that these metrics are mostly statistically independent and, therefore, do not capture a great deal of redundant information."

Source: Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, 1996. doi:10.1109/32.544352


Towards a Metrics Suite for Object Oriented Design
Chris F Kemerer
Shyam R Chidamber
Sloan School of Management. Center for Information Systems Research

"A small number of classes may be responsible for a large number of the methods that executed in an application, and that if testing effort were concentrated on these outlier classes, a bulk of the dynamic behavior of the object oriented systems can be checked for customer acceptance."

— Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994. doi:10.1109/32.295895

Mark Lorenz

"Number of Messages (NOM) measures the number of messages sent in a method, segregated by type of message."

— Mark Lorenz and Jeff Kidd. *Object-Oriented Software Metrics: A Practical Guide.* Prentice-Hall, Inc, 1994. doi:10.5555/177063

Number of Messages (NOM)

```
class Trim
  String value()

class Book
  public String title()
    return new Trim(
      this.read()
    ).value();
  private String read()
    // read from the DB
```

$\texttt{NOM}_{\texttt{title}} = 3$

"Number of Messages (NOM) measures the number of messages sent in a method, segregated by type of message."

References

Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, 1996. doi:10.1109/32.544352.

Fernando Brito and Rogério dos Santos Carapuça. Object-Oriented Software Engineering: Measuring and Controlling the Development Process. In *Proceedings of the 4th International Conference on Software Quality*, pages 1–8, 1994.

Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6): 476–493, 1994. doi:10.1109/32.295895.

Rachel Harrison, Steve J. Counsell, and Reuben V. Nithi. An Evaluation of the MOOD Set of Object-Oriented Software Metrics. *IEEE Transactions on Software Engineering*, 24(6): 491–496, 1998. doi:10.1109/32.689404.

Mark Lorenz and Jeff Kidd. *Object-Oriented Software Metrics: A Practical Guide.* Prentice-Hall, Inc, 1994. doi:10.5555/177063.