

# Object Dimensions

YEGOR BUGAYENKO

Lecture #10 out of 24

80 minutes

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as website. Copyright belongs to their respected authors.

## 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)

## Metrics for Object-Oriented Design (MOOD)

+2pt



“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.”

— *Object-oriented software engineering: Measuring and controlling the development process*, Fernando Brito e Abreu and Rogério Carapuça, Proceedings of the 4th International Conference on Software Quality, 1994

## Method Hiding Factor (MHF)

```
1 class Book {  
2     public String title() {  
3         return this.read() + ".";  
4     }  
5     private String read() {  
6         // read from database  
7     }  
8 }
```

$$\text{MHF} = 1/2$$

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

## Attribute Hiding Factor (AHF)

```
1 class Book {  
2     public int id;  
3     private File path;  
4     public String title() {  
5         var txt = Files.read(path);  
6         // 1. Find title by 'id'  
7         // 2. Return it  
8     }  
9 }  
10  
11 var b = new Book();  
12 b.id = 42;  
13 var t = b.title();
```

$$\text{AHF} = 1/2$$

“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)

```
1 class Book extends Material {  
2     @Override  
3     public String content() {  
4         return "Hello, world!";  
5     }  
6     public String title() {  
7         return "David West";  
8     }  
9 }
```

$MIF = 1/2$

“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)

```
1 class Material {  
2     protected String content;  
3 }  
4  
5 class Book extends Material {  
6     private String title;  
7 }
```

$$\text{AIF} = 1/2$$

“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)

```
1 class Material {  
2     public String content() {  
3         return "Hello, world!";  
4     }  
5 }  
6  
7 class Book extends Material {  
8     public String title() {  
9         return "David";  
10    }  
11 }
```

$PF = 0/1$

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



+2pt



“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, 1994

## Depth of Inheritance Tree (DIT)

```
1 class Material {  
2     public String content() {  
3         return "Hello, world!";  
4     }  
5 }  
6  
7 class Book extends Material {  
8     public String title() {  
9         return "David";  
10    }  
11 }
```

$DIT_{\text{Material}} = 0$

$DIT_{\text{Book}} = 1$

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

## Number of Children (NOC)

```
1 class Material {  
2     public String content() {  
3         return "Hello, world!";  
4     }  
5 }  
6  
7 class Book extends Material {  
8     public String title() {  
9         return "David";  
10    }  
11 }
```

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

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

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

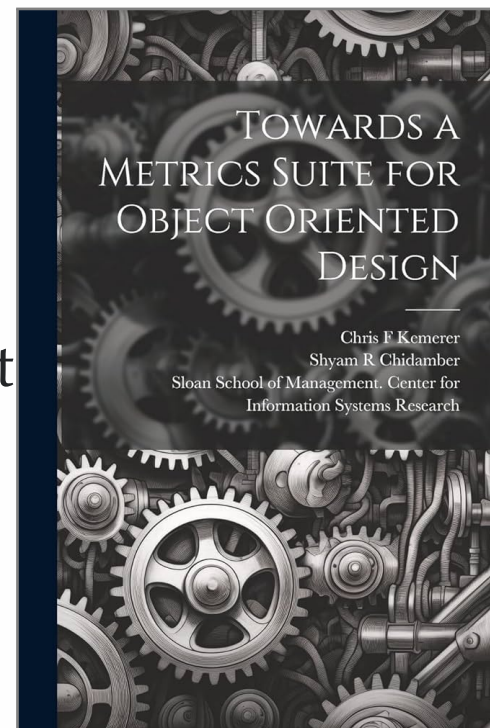
## Response For a Class (RFC)

```
1 class Trim
2   String value()
3
4 class Book
5   public String title()
6     return new Trim(
7       this.read()
8     ).value();
9   private String read()
10    // read from the DB
```

$$\text{RFC}_{\text{Material}} = 3$$

“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.”

+2pt



“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, 1994

+2pt



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

— *Object-Oriented Software Metrics: a Practical Guide*,  
Mark Lorenz and Jeff Kidd, Prentice-Hall, Inc., 1994

## Number of Messages (NOM)

```
1 class Trim
2     String value()
3
4 class Book
5     public String title()
6         return new Trim(
7             this.read()
8             ).value();
9     private String read()
10        // read from the DB
```

$NOM_{title} = 3$

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

## Typical Mistakes in Research Papers

1. Assuming the reader understands the context
2. Trying to find impact instead of correlation
3. Ignoring the importance of taxonomy in Related Work
4. Writing in future tense



## Read this:

*Evaluating the Impact of Object-Oriented Design on Software Quality*, Fernando Brito e Abreu and Walcélio Melo, Proceedings of the 3rd International Software Metrics Symposium (METRICS'96), IEEE, 1996

*An Overview of Object-Oriented Design Metrics*, Daniel Rodriguez and Rachel Harrison, The University of Reading, 2001

*Object-Oriented Software Metrics: a Practical Guide*, Mark Lorenz and Jeff Kidd, Prentice-Hall, Inc., 1994

[OOP Alternative to Utility Classes](#) (2014)

[Java Annotations Are a Big Mistake](#) (2016)

[How Much Your Objects Encapsulate?](#) (2014)