# Coupling

#### YEGOR BUGAYENKO

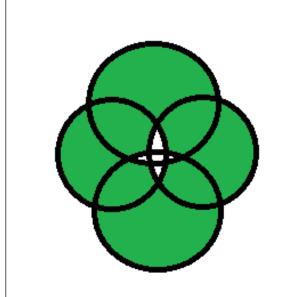
# Lecture #6 out of 24 80 minutes

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as website.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



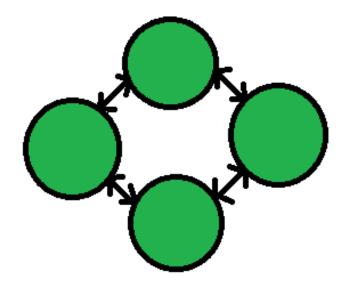
"The fewer and simpler the connections between modules, the easier it is to understand each module without reference to other modules."

Wayne P. Stevens, Glenford J. Myers, and <u>Larry L.</u>
 <u>Constantine</u>, *Structured Design*, IBM Systems Journal 13.2 (1974)



### Tight coupling:

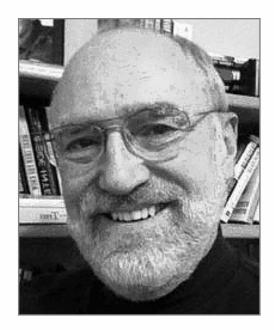
- 1. More Interdependency
- 2. More coordination
- 3. More information flow



### Loose coupling:

- 1. Less Interdependency
- 2. Less coordination
- 3. Less information flow

Source: https://www.geeksforgeeks.org/coupling-in-java/



"Coupling is the measure of the strength of association established by a connection from one module to another. Strong coupling complicates a system since a module is harder to understand, change, or correct by itself if it is highly interrelated with other modules. Complexity can be reduced by designing systems with the weakest possible coupling between modules."

Wayne P. Stevens, <u>Glenford J. Myers</u>, and Larry L.
Constantine, *Structured Design*, IBM Systems Journal 13.2 (1974)



Source: https://www.javatpoint.com/software-engineering-coupling-and-cohesion



"The degree of coupling established by a particular connection is a function of several factors, and thus it is difficult to establish a simple index of coupling. Coupling depends (1) on how complicated the connection is, (2) on whether the connection refers to the module itself or something inside it, and (3) on what is being sent or received."

Wayne P. Stevens, Glenford J. Myers, and Larry L.
 Constantine, *Structured Design*, IBM Systems Journal 13.2 (1974)



Source: https://nordicapis.com/the-difference-between-tight-coupling-and-loose-coupling/

\*\*Coupling Between Objects (CBO) — for a class is a Coupling Between Objects (CBO) — for a class

Suite for Object Oriented Design, IEEE Transactions on Software Engineering, 20.6, 1994

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 1, JANUARY

#### A Hierarchical Model for Object-Oriented Design Quality Assessment

agdish Bansiya, Member, IEEE, and Carl G. Davis, Fellow, IEEE

dex Terms—Quality model, quality attributes, design metrics, product metrics, object-oriented metrics

#### INTRODUCTIO

1 Introduction
In demand for quality software continues to intensity
If the too our society's increasing dependence on software and the other develowing effect that a software error can add the other develower error can obtain a software systems must ensure consistent and error first software systems must ensure consistent and error first operation every time they are used. This demand for increased software quality has resulted in quality being control error of adifferentiate between products have it even to the control of the error of the

The switch to the object-oriented parally are far from sected stocked for the switch to the object-oriented parallegin has charged the elements that we not object-oriented parallegin flass of the switching object or switching object or switching object oriented paralleging the switching object operations such as encapsulation, inheritance, an object-orientation to object-orientation to object-orientation for the switching object oriented to the switching object of the object-orientation of the switching object of the object-orientation of the switching object orientation of the object-orientation of the object-orientatio

However, the new object-oriented metrics are varied in what they measure, how they are used in measuring, an when they are applicable. Many of the newer metrics have only been validated with small, and sometimes nonrealisti-

- J. Bansiya is with the Department of Mathematics and Comput California State University, Hoyanard, CA 94542.
   E-wall: jhansiya@csuksyyoord.afu.
- C. Durs is with the Conyader Science Department, University of Alabam in Huntsville, Huntsville, Al. 35598; E-mail: classifies sait afu.
   Manuscript received 24 Nov. 1597; revised 29 Nov. 1598; accepted 27 Jan 2000.
   Romstronded for accontance by D.R. Jeffres.

ommended for acceptance by D.R. Jeffery.
information on obtaining reprints of this article, please send e-mail:
Becommuter.org. and reference IEEE/CS Los Namber 169978.

data sets and, therefore, the practical applicability and effectiveness of the metrics on large complex projects such as those encountered in an industrial environment is not known. Finally, if the goal is assessing the external quality attributes of the product rather than simply collecting individual metrics, then there must be a well defined way of

study of no interiors and quality modes. Certain applied only after a product in complete, or nearly complete. They rely upon information extracted from the implementation of the product. This provides information too late to help in improving internal product archaertesit too late to help in improving internal product characteristic too late to help in improving internal product characteristic too late to help in the product. This works in the real as necessary of the product that the analysis and design howe forwards internal proper tites that will lead to the development of a quality or product. This works disprinted to employed the product that the analysis and design howe forwards only help in the product. The work of the product that the analysis and development of the mission of the product that the analysis and development of the size of the product that the analysis and the product that the product tha

and resource planning, under the property of t

0088-559902517.00 © 2002 IEEE
thorized licensed use limited to: ECOLE POLYTECHNIQUE DE MONTREAL Downloaded on October 7, 2009 at 11:08 from IEEE Xplore. Restrictions of

"Direct Class Coupling (DCC) — this metric is a count of the different number of classes that a class is directly related to. The metric includes classes that are directly related by attribute declarations and message passing (parameters) in methods."

 Jagdish Bansiya and Carl G. Davis, A Hierarchical Model for Object-Oriented Design Quality Assessment,
 IEEE Transactions on Software Engineering, 28.1, 2002



"The biggest problems come from uncontrolled coupling at the <u>upper levels</u>. I don't worry about the number of modules coupled together, but I look at the pattern of dependency relationship between the modules."

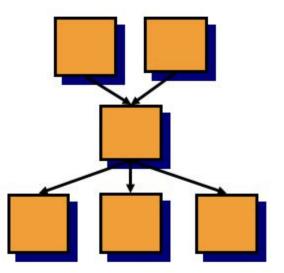
- Martin Fowler, *Reducing Coupling*, IEEE Software, 2001



"Low-to-medium fan-out means having a given class use a low-to-medium number of other classes. High fan-out (more than about seven) indicates that a class uses a large number of other classes and may therefore be overly complex. High fan-in refers to having a high number of classes that use a given class. High fan-in implies that a system has been designed to make good use of utility classes at the lower levels in the system."

Steven McConnell, Code Complete, 2004

Fan-in = number of ingoing dependencies Fan-out = number of outgoing dependencies



Heuristic: a high fan-in/fan-out indicates a high complexity

(c) Natalia Kokash, Leiden Institute of Advanced Computer Science

#### An Evolutionary Study of Fan-in and Fan-out Metrics in OSS

A. Mubarak, S. Counsell and R.M. Hierons lepartment of Information Systems and Computing, Brunel University Uxbridge, UK. Email: steve.counsell@brunel.ac.uk

Abstrace Exercise coupling between object-oriented classes in which packnowledge an animateness problem that can read it a bight packnowledge and animateness problem that can read it a bight packnowledge and the bight packnowledge and the confidence of the confide

eywords-coupling, Java, fan-in, fan-out, package.

recognition of mains in contrasted 1), it is worse, releases to every different production of the complexity that or complicate subsequent maintenance and represents a victor of complexity that or complicate subsequent maintenance and represents a victor of complexity of the comple

in this paper, we investigate versions of five Open Source systems (OSS) Sociation on two well-known coupling metrics "fair-in' (i.e., incoming coupling) and "fan-out" (i.e., otagoing coupling). We used an automated tool to extract each of the coupling metrics from those five systems. The research puestions we explore are first, is in the cast that classes with arge incoming coupling naturally have low outgoing coupling and second, does this relationship worsen over time? In other words, does the potential maintenance problem become worse not more of fair-in and fanou values?

II. MOTIVATION AND RELATED WORK

The reason is nith paper in motivated by a number of factors. Firstly, periods reasons [15] has shown that there is a tradeoff between coupling types — is particular, that between coupling the particular of the particular of the particular internation-less particular coupling in this paper, we explore the potential characteristics and trade-offs between fari- and faces are more control of the particular coupling in the particular developers through techniques such as refactoring [17] and however, the practical casellation of limited international control of the particular coupling to become 'smallier' and, if is, in a dead with in this paper, we explore, nor time modested by such as the paper of the particular coupling to become 'smallier' and, if is, in a dead with in this paper, we explore, nor time modested by our motive studied to be relatively small for clease removed on a motive studied to be relatively small for clease removed moder fancous may be difficult to move or remove from a confer fancous may be difficult to move or remove from a treatment of the particular control of the control of the particular control in the two mortest controlled.

In terms of related work, the research presented relates to areas of software evolution, coupling metrics and the use of OSS [8]. In terms of software evolution, the laws of Lehrama [2] provide the backdrop for many past evolutionary studies. Evolution has also been the subject of simulation studies [18] and this has allowed OSS evolution to be studied in a contrasting way to that empirically. The research presented in

"We also found evidence of certain 'key' classes (with both high fan-in and fan-out) and 'client' and 'server'-type classes with just high fan-out and fan-in, respectively."

A. Mubarak et al., An Evolutionary Study of Fan-in and Fan-out Metrics in OSS, Proceedings of the 4th
 International Conference on Research Challenges in
 Information Science (RCIS), 2010

Fan-out, as a metric, is supported by a few tools:

• Checkstyle for Java

 $\bullet$  <u>CCCC</u> for C++, C, and Java

• module-coupling-metrics for Python



"Afferent coupling (denoted by  $\mathbf{Ca}$ ) is a metric that indicates the total number of other projects/boundaries that are dependent upon it. Efferent coupling (denoted by  $\mathbf{Ce}$ ) is another metric that is the verse of Afferent Coupling. It is the total number of projects that a given project depends on. Instability another metric that is a ratio:  $\mathbf{I} = \mathbf{Ce}/(\mathbf{Ce} + \mathbf{Ca})$ . This metric is a ratio between 0 and 1. With 0 meaning it's totally stable and 1 meaning it's unstable."

 Derek Comartin, Write Stable Code using Coupling Metrics, 2021

# Types of Coupling (some of them)

- Content Coupling is when one module modifies or relies on the internal workings of another module (e.g., accessing local data of another module).
- Global Coupling is when two modules share the same global data (e.g., a global variable).
- External Coupling occurs when two modules share an externally imposed data format, communication protocol, or device interface.
- <u>Control Coupling</u> is one module controlling the flow of another, by passing it information on what to do (e.g., passing a what-to-do flag).
- Stamp Coupling is when modules share a composite data structure and use only a part of it, possibly a different part (e.g., passing a whole record to a function that only needs one field of it).

- <u>Data Coupling</u> is when modules share data through, for example, parameters. Each datum is an elementary piece, and these are the only data shared (e.g., passing an integer to a function that computes a square root).
- Message Coupling can be achieved by state decentralization (as in objects) and component communication is done via parameters or message passing (see Message passing).
- <u>Subclass Coupling</u> describes the relationship between a child and its parent. The child is connected to its parent, but the parent isn't connected to the child.
- Temporal Coupling is when two actions are bundled together into one module just because they happen to occur at the same time.

#### Source:

https://wiki.edunitas.com/IT/en/114-10/Coupling-(computer-programming)\_1430\_eduNitas.html

# Fear of Decoupling

```
interface Money {
  double cents();
}

void send(Money m) {
  double c = m.cents();
  // Send them over via the API...
}

class OneDollar implements Money {
  @Override
  double cents() {
    return 100.0d;
  }
}
```

```
class EmployeeHourlyRate
implements Money {
    @Override
    double cents() {
        // Fetch the exchange rate;
        // Update the database;
        // Calculate the hourly rate;
        // Return the value.
    }
}
```

"Polymorphism makes sofware more fragile ... to make it more robust!"

# Temporal Coupling

### Tight coupling (not good):

```
List<String> list =
new LinkedList<>();
Foo.append(list, "Jeff");
Foo.append(list, "Walter");
return list;
```

### Loose coupling (good):

```
return Foo.with(
Foo.with(
new LinkedList<>(),
"Jeff"
),
"Walter"
);
```

https://www.yegor256.com/2015/12/08/temporal-coupling-between-method-calls.html

# Distance of Coupling

```
class Temperature {
  private int t;
  public String toString() {
    return String.format("%d F", this.t);
  }
}

Temperature x = new Temperature();
String txt = x.toString();
String[] parts = txt.split(" ");
int t = Integer.parseInt(parts[0]);
```

"The larger the number (or the mean of all numbers), the worse the design: in good design we are not supposed to take something out of a method and then do some complex processing. The distance metric will tell us exactly that: how many times, and by how much, we violated the principle of loose coupling."

https://www.yegor256.com/2020/10/27/distance-of-coupling.html

## Read this:

Structured Design, Wayne P. Stevens, et al., IBM Systems Journal, 13.2, 1974

A Hierarchical Model for Object-Oriented Design Quality Assessment, Jagdish Bansiya et al., IEEE Transactions on Software Engineering, 28.1, 2022

An Overview of Various Object Oriented Metrics, Brij Mohan Goel et al., International Journal of Information Technology & Systems, 2.1, 2014

Analysing the Contribution of Coupling Metrics for the Development and Management of Process Architectures, Daniel Braunnagel et al., ECIS, 2015

New Metric: the Distance of Coupling (2020)

Fear of Decoupling (2018)

Reflection Means Hidden Coupling (2022)