

YEGOR BUGAYENKO

Lecture #6 out of 24 80 minutes

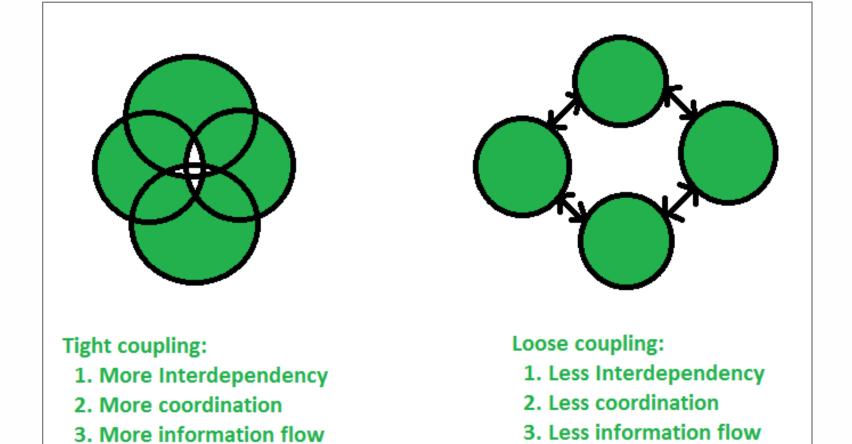
The slidedeck was presented by the author in this YouTube Video

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as web sites.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

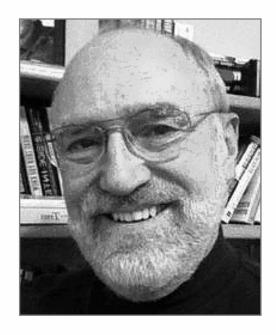


"The fewer and simpler the connections between modules, the easier it is to understand each module without reference to other modules."

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



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 Months Suite for Object Oriented Design And Metrics Suite for Object Oriented Design And For Metric Suite for Object Oriented Design And For Object Oriented Des

Software Engineering, 20.6, 1994

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

"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

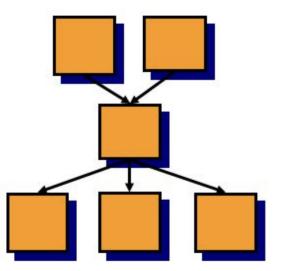


STEVE McConnell

"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."

— Steve McConnell. Code Complete. Pearson Education, 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
epartment of Information Systems and Computing, Brunel University

Lybridge LIK Email: stope counsell@beunel as uk

Admires. Executive copyling between object-oriented clauses with peak-oriented and a maintenance problem that are result with peak-oriented maintenance problem. The sim of this paper is to epitere the relational towards. The sim of this paper is to epitere the relational towards of the simulation of the simulation over mining provide many or the control of the simulation over mining provide makes one to conserve filling. The allows from what the relationship between the two matrices the determining pattern of the simulation of the simulation of the control filling. The allows from what the relationship is the control filling the filling which the simulation of the control filling the filling the simulation of the control filling the filling which the simulation of the companion and existence of such datasets with path places and fall and evidence of such datasets. The simulation of the companion and existence of such datasets and participated on the companion and existence of such datasets.

Ceywords-coupling, Java, fan-in, fan-out, package

I. INTRODUCTION

Executive disse coupling has often been related to the the Object-Oriented (co) community that executive coupling between classes creates a level of complexity that can be the object-Oriented (co) community that executive coupling between classes creates a level of complexity that can be completed to the complete complete complete control or many other classes is in ideal condidate for reaching the complete control or many other classes is in ideal condidate for the complete complete control or control oriented classes with high condiging in the control oriented complete control oriented complete control oriented control orient

n this paper, we investigate versions of five Open Source systems (OSS) Gossing on two well-known coupling metrics 'fain-in' (i.e., incoming coupling) and 'fain-out' (i.e., outgoing outpiling). We used an automated tool to extract each of the oupling metrics from those five systems. The research to the outpiling in the outpiling in the outpiling of the outpiling in the outpiling in the outpiling in the outpiling in the outpiling outpiling and second, does this relationship worsen over time? In other ords, does the potential maintenance problem become worse

II. MOTIVATION AND RELATED WORK

The reason's nit sha page in motivated by a number of finetics of the Section of the Section Could be seen as the section of the Section Could be seen as the Section of the Section Could be seen as the Section of the Section Could be seen as the Section of the Section S

In terms of related work, the research presented relates to areas of software evolution, coupling metries and the use of OSS [8]. In terms of software evolution, the laws of Lehman [2] provide the backdrop for many past evolutionary studies. Production has also been the subject of simulation studies [18] Evolution has also been the subject of simulation studies [18] contenting many to that empirically. The research persented in this name, globys, into weap[16, 20].

"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.
- <u>Temporal Coupling</u> 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"
);
```

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)

References

Steve McConnell. *Code Complete*. Pearson Education, 2004.