

YEGOR BUGAYENKO

Lecture #6 out of 24 80 minutes

The slidedeck was presented by the author in this YouTube Video

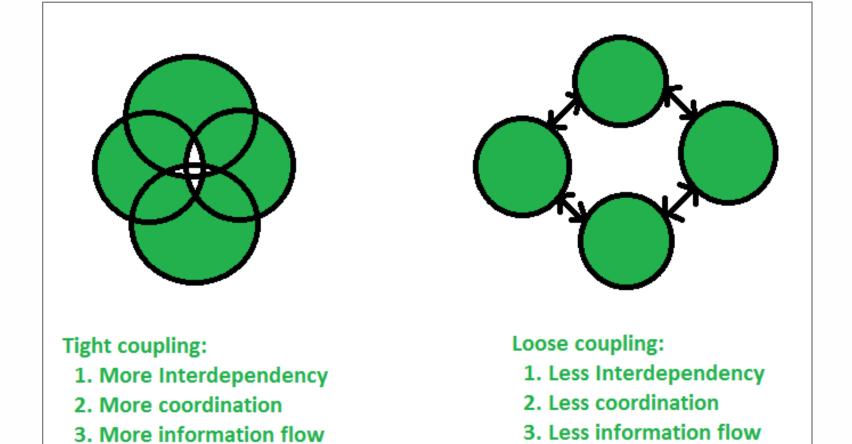
All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as web sites. Copyright belongs to their respected authors.



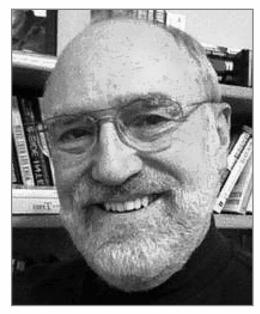
Wayne P. Stevens, Glenford J. Myers, and Larry L. Constantine

"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 Larry L. Constantine. Structured Design. *IBM Systems Journal*, 13(2):115–139, 1974



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



GLENFORD MYERS

"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, Glenford J. Myers, and Larry L. Constantine. Structured Design. *IBM Systems Journal*, 13(2):115–139, 1974



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



WAYNE P. STEVENS

"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):115–139, 1974



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

A Metrics Suite for Object Oriented Design coupled." — Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object

"Coupling Between Objects (CBO) — for a class is a count of the number of other classes to which it is

Oriented Design. IEEE Transactions on Software Engineering, 20(6):476-493, 1994

A Hierarchical Model for Object-Oriented Design Quality Assessment Jagdish Bansiya, Member, IEEE, and Carl G. Davis, Fellow, IEEE

Abstract—This paper describes an improved freementule model for the assessment of high level design quality articles an elegaciontered designs. In the model, muchal and behavior design properties of lesses, ediques, and the interiorings are enablead using a site of biject owner designs prefets. The model relative design properties such as exequitation, muchanity, copying, and relative to the control of the properties of the properties of the an exequitation, muchanity, copying, and relative to the properties of the properties

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

INTRODUCTION

The demand for quality software continues to intensity due to tour society's increasing dependence on orderest and the other devestating effect that a software error can all the other devestating effect that a software error can be recommended to the control of the control of

measure and assure quality are far from settled issues. The switch to the object-criented paradigm has change the elements that we use to assess software qualit Traditional software product metrics have evaluate producharacteristics such as sixe, complexity, performance, are different notions such as encapsulation, inheritance, are polymorphism which are inherent in object-orientation This has led to the definition of namy new metrics [8], [12] [20] to measure the products of the object-orienta approach.

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

- California State University, Haynaini, CA 94542.
 E-mail: jbauxiquificulusyssood afu.

 C. Davis is with the Computer Science Department, University of Alaban in Hantsville, Huntsville, Al. 35893. E-mail: advantiles ush afu.
- C. Danis is with the Computer Science Department, University of Alebon in Huntstriptifite, Huntrallile, AI, 35593. E-mail: calconition, until only Manuscript received 24 Nov. 1997; revised 29 Nov. 1998; accepted 27 Ja 2000.
 Recommoded for acceptance by D. R. Joffers.

0098-5599/02/517.00 © 2002 IEEE
uthorized licensed use limited to: ECOLE POLYTECHNIQUE DE MONTREAL Downloaded on October 7, 2009 at 11:05 fn

effectiveness of the metrics on large complex projects su as those encountered in an industrial environment is n known. Finally, if the goal is assessing the external quali attributes of the product rather than simply collectiindividual metrics, then there must be a well defined way connecting the two.

Many of the metrics and quality models currently available for object-oriented software analyses can be available for object-oriented software analyses can be complete. They rely upon information extracted from the implementation of the product. This provides information too late to help in improving internal product characteristic typic to the complete of the product. This provides information to the product of the produc

Fortunately, the object-oriented approach naturally intent for no early assessment and evaluation. Object-oriented for a nearly assessment and evaluation. Object-oriented for a new particular oriented and programment type to identify objects and clauses, attribute an operations, and relationships. Encapatation, inheritance and polymorphism require deligenes to carefully structure delarges and content of the intention theorem objects. The design and comment the intention theorem objects in for implementation. Therefore, the approach provides the information needed to assess the quality of a design't clause, structure, and relationships before they are committed to an implementation.

Authorized Icensed use limited to: ECOLE POLYTECHNIQUE DE MONTREAL. Downloaded on October 7, 2009 at 11:09 from IEEE Xplore. Restrictions apply.

Jagdish Bansiya and Carl G. Davis "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."

— J. Bansiya and C.G. Davis. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering*, 2002. doi:10.1109/32.979986



Martin Fowler

"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."

— M. Fowler. Reducing coupling. *IEEE Software*, 2001. doi:10.1109/ms.2001.936226

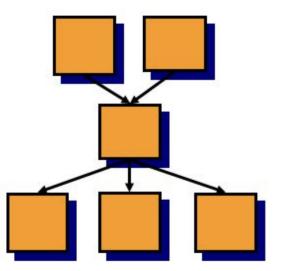


STEVE McConnell

"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

Department of Information Systems and Computing, Brunel University

Lybridge LIK Empil steep counsell@beunel.ac.uk

Abonce. Exercise copying between object-releated clause is a skipler proposate for facilities varieties and street of principles of the property of the proper

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

oupling has often been

the Object-Oriented (OO) community that excessive cought between classes exercise a level of completely that car complicate subsequent maintenance and represents a Victor complicate subsequent maintenance and represents a Victor complete through the classes it as in ideal candidate for reception of the complete complete through the consideration of the complete comple this paper, we investigate versions of five Open Source, system (CSS) focusing on two well-known coughing metrics from the control of the c

I. MOTIVATION AND RELATED WORK

Findly, previous resized [15] has above that there is a tradeoff between coupling types — is particular, that between coupling through imported packages and the introduction of the potential characteristic and trade-off-between fine-in and fine-cut marties over time. Second, was would always expected by the potential characteristic and trade-off-between fine-in and fine-cut marties over time. Second, was would always expecded-overlyine, through the characteristic [16], however, the practical relative thereigness and resources at particularly had wrateful (e.g. excessive coupling) [19] are they dealt with, I not in paper, we explore, one time, whether smalls in the contract of the contraction of the contraction of the waste propertions. I model used to the contraction of the waste propertions. I model used to the contraction of the

In terms of related work, the research presented relates to areas of software evolution, coupling metrics and the use of OSS B[]. In terms of software evolution, the laws of Lehman [2] provide the backdrop for many past evolutionary studies [18] 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, S. Counsell, and R.M. Hierons. An evolutionary study of fan-in and fan-out metrics in OSS. In *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2010. doi:10.1109/RCIS.2010.5507329

A. Mubarak et al.

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



DEREK COMARTIN

"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."

//codeopinion.com/write-stable-code-using-coupling-metrics/

[—] Derek Comartin. Write Stable Code using Coupling Metrics, 2021. URL https:

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

- J. Bansiya and C.G. Davis. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering*, 2002. doi:10.1109/32.979986.
- Shyam R. Chidamber and Chris F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6): 476–493, 1994.
- Derek Comartin. Write Stable Code using Coupling Metrics, 2021. URL https://codeopinion.com/

- write-stable-code-using-coupling-metrics/.
- M. Fowler. Reducing coupling. *IEEE Software*, 2001. doi:10.1109/ms.2001.936226.
- Steve McConnell. *Code Complete*. Pearson Education, 2004.
- A. Mubarak, S. Counsell, and R.M. Hierons. An evolutionary study of fan-in and fan-out metrics in OSS. In *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2010. doi:10.1109/RCIS.2010.5507329.
- Wayne P. Stevens, Glenford J. Myers, and Larry L. Constantine. Structured Design. *IBM Systems Journal*, 13(2):115–139, 1974.