TCC and LCC
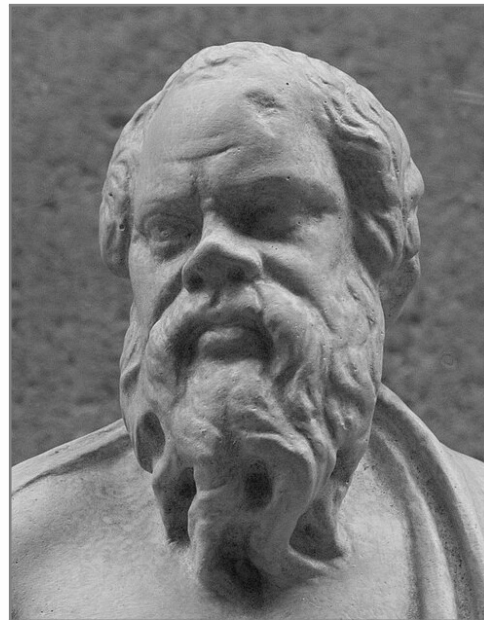
Yegor Bugayenko

Lecture #8 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

"**Socrates**: I am a lover of these processes of <u>division</u> and <u>bringing together</u>, as aids to speech and thought; and if I think any other man is able to see things that can naturally be collected into one and divided into many, him I follow after and walk in his footsteps as if he were a god."

— Plato. Phaedrus (Dialogue), 370 B.C.

Edward Yourdon

"Module cohesion may be conceptualized as the cement that holds the processing elements of a module together. In a sense, a high degree of module cohesion is an indication of close approximation of inherent problem structure."

— Edward Yourdon and Larry Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice Hall, 1979. doi:10.5555/578522
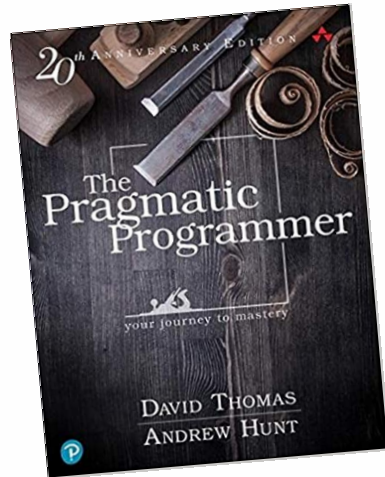
"We want to design components that are self-contained: independent, and with a single, well-defined purpose."

— Andrew Hunt and Dave Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Pearson Education, 1999. doi:10.5555/320326

ANDREW HUNT
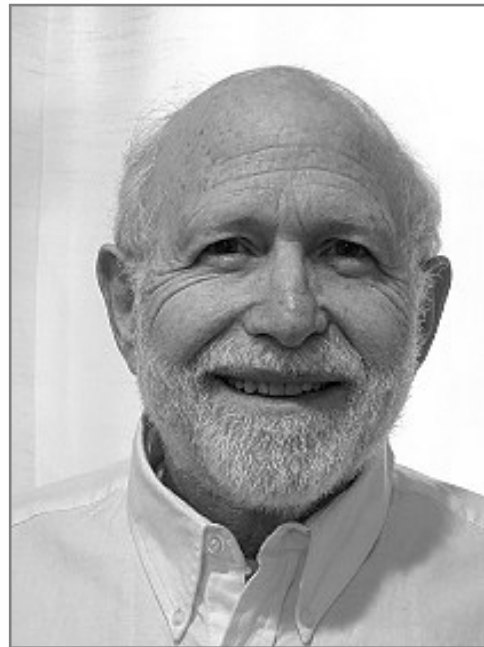
JAMES M. BIEMAN

"We define two measures of class cohesion based on the <u>direct</u> and <u>indirect</u> connections of method pairs: TCC and LCC."

— James M. Bieman and Byung-Kyoo Kang. Cohesion and Reuse in an Object-Oriented System. *SIGSOFT Software Engineering Notes*, 20(51):259–262, 1995. doi:<u>10.1145/223427.211856</u>

Connectivity Between Methods in a Class

3.1 Connectivity between methods

The direct connectivity between methods is determined from the class abstraction. If there exists one or more common instance variables between two method abstractions then the two corresponding methods are *directly connected*.

Two methods that are connected through other directly connected methods are *indirectly connected*. The indirect connection relation is the transitive closure of direct connection relation. Thus, a method M_1 is indirectly connected with a method M_n if there is a sequence of methods $M_2, M_3, \ldots, M_{n-1}$ such that

$$M_1 \, \delta \, M_2, \cdots, M_{n-1} \, \delta \, M_n$$

where $M_i \, \delta \, M_j$ represents a direct connection.

"An instance variable is directly used by a method M if the instance variable appears as a data token in the method M. The instance variable may be defined in the same class as M or in an ancestor class of the class. $DU(M)$ is a set of instantce variables directly used by a method M."

Source: James M. Bieman and Byung-Kyoo Kang. Cohesion and Reuse in an Object-Oriented System. *SIGSOFT Software Engineering Notes*, 20(51):259–262, 1995. doi:10.1145/223427.211856

Tight and Loose Class Cohesion (TCC+LCC)

```
1  class Rectangle
2    int x, y, w, h;
3    int area()
4      return w * h;
5    int move(int dx, dy)
6      x += dx; y += dy;
7    int resize(int dx, dy)
8      w += dx; h += dy;
9    bool fit()
10     return w < 100
11       && x < 100;
```

Max possible connections ($\texttt{NP}$):
$N \times (N - 1)/2 = 4 \times 3/2 = 6$

Directly connected ($\texttt{NDC} = 4$):
area+fit, area+resize, move+fit, resize+fit

Indirectly connected ($\texttt{NIC} = 2$):
area+move, move+resize

$\texttt{TCC} = \texttt{NDC}/\texttt{NP} = 4/6 = 0.66$
$\texttt{LCC} = (\texttt{NDC}+\texttt{NIC})/\texttt{NP} = 6/6 = 1.00$

"If a class is designed in <u>ad hoc manner</u> and unrelated components are included in the class, the class represents more than one concept and does not <u>model an entity</u>. The cohesion value of such a class is likely to be less than 0.5."

— James M. Bieman and Byung-Kyoo Kang. Cohesion and Reuse in an Object-Oriented System. *SIGSOFT Software Engineering Notes*, 20(51):259–262, 1995. doi:<u>10.1145/223427.211856</u>
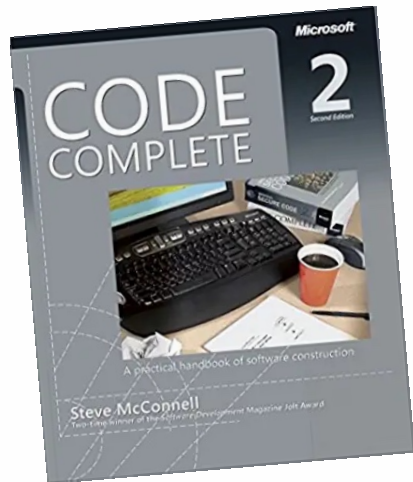
Steve McConnell

"Cohesion refers to how closely all the routines in a class or all the code in a routine support a central purpose—how focused the class is. The ideas of abstraction and cohesion are closely related—a class interface that presents a good abstraction usually has strong cohesion."

— Steve McConnell. *Code Complete.* Pearson Education, 2004. doi:10.5555/1096143

Abstraction
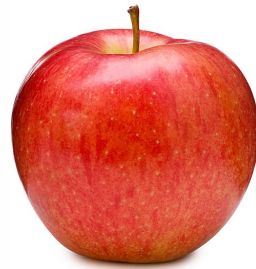
- Color: red

- Weight: 120g

- Price: $0.99

The slide is taken from the "Pain of OOP" (2023) course.

```
1  var file = {
2    path: '/tmp/data.txt',
3    read: function() { ... },
4    write: function(txt) { ... }
5  }
```
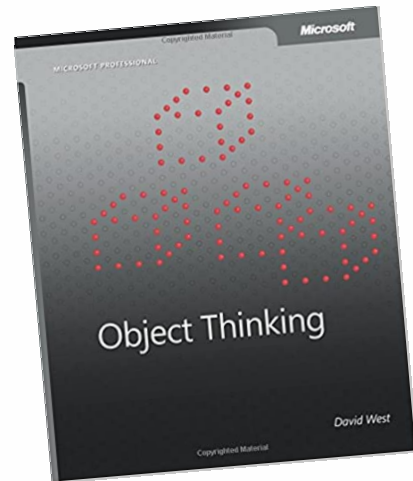
"Isomorphism of the modules (objects) in problem and solution space is a desirable, in fact essential, quality for software."

— David West. *Object Thinking*. Pearson Education, 2004. doi:10.5555/984130
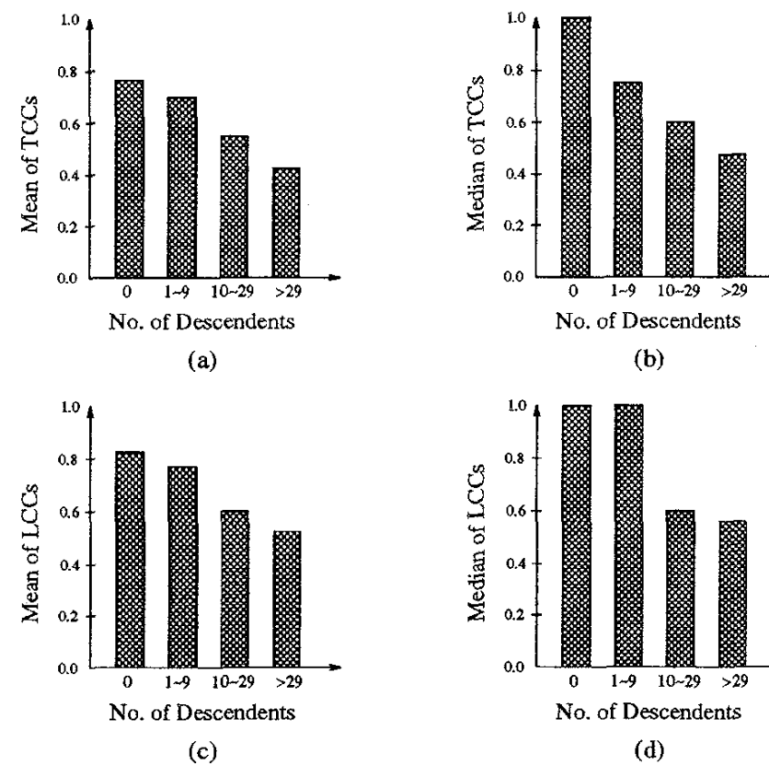
Inheritance vs. Cohesion

Figure 3: Number of descendents and Class Cohesion

"Our results show that the classes that are heavily <u>reused</u> via inheritance exhibit <u>lower cohesion.</u> We expected to find that the most reused classes would be the most cohesive ones."

Inheritance is Code Reuse

```
1  class Manuscript {
2    protected String body;
3    void print(Console console) {
4      console.println(this.body);
5    }
6  }
7  class Article
8    extends Manuscript {
9    void submit(Conference cnf) {
10     cnf.send(this.body);
11   }
12 }
```

"The `Article` copies method `print()` and attribute body from the `Manuscript`, as if it's not a living organism, but rather a dead one from which we <u>inherit</u> its parts."

"Implementation inheritance was created as a mechanism for code reuse. It doesn't fit into OOP at all."

Source: Yegor Bugayenko. Inheritance Is a Procedural Technique for Code Reuse. https://www.yegor256.com/160913.html, sep 2016. [Online; accessed 22-09-2024]

Composition over Inheritance

```
1  class Manuscript
2    protected String body;
3    void print(Console console)
4      console.println(this.body);
5
6  class Article
7    extends Manuscript
8    void submit(Conference cnf)
9      cnf.send(this.body);
```

```
1  class Manuscript
2    protected String body;
3    void print(Console console)
4      console.println(this.body);
5
6  class Article
7    Manuscript manuscript;
8    Article(Manuscript m)
9      this.manuscript = m;
10   void submit(Conference cnf)
11     cnf.send(this.body);
```

Wikipedia: https://en.wikipedia.org/wiki/Composition_over_inheritance

TCC+LCC can be calculated by a few tools:

- jPeek for Java

- C++ — don't know

- Python — don't know

- JavaScript — don't know

- C# — don't know

References

James M. Bieman and Byung-Kyoo Kang. Cohesion and Reuse in an Object-Oriented System. *SIGSOFT Software Engineering Notes*, 20(51): 259–262, 1995. doi:10.1145/223427.211856.

Yegor Bugayenko. Inheritance Is a Procedural Technique for Code Reuse. https://www.yegor256.com/160913.html, sep 2016. [Online; accessed 22-09-2024].

Andrew Hunt and Dave Thomas. *The Pragmatic Programmer: From Journeyman to Master.* Pearson Education, 1999. doi:10.5555/320326.

Steve McConnell. *Code Complete.* Pearson Education, 2004. doi:10.5555/1096143.

Plato. Phaedrus (Dialogue), 370 B.C.

David West. *Object Thinking.* Pearson Education, 2004. doi:10.5555/984130.

Edward Yourdon and Larry Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice Hall, 1979. doi:10.5555/578522.