TCC and LCC

Yegor Bugayenko

Lecture #8 out of 24

80 minutes

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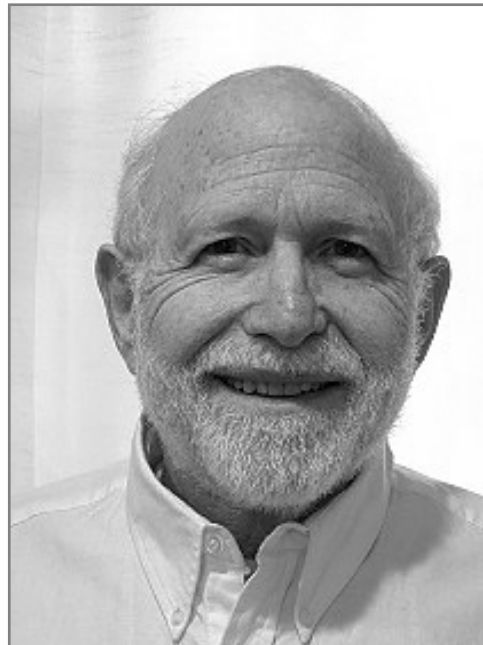
"We define two measures of class cohesion based on the direct and indirect connections of method pairs: TCC and LCC."

— James M. Bieman and Byung-Kyoo Kang, *Cohesion and Reuse in an Object-Oriented System*, Proceedings of the Symposium on Software Reusability (SSR), 1995

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 (NP):
$$N \times (N-1)/2 = 4 \times 3/2 = 6$$

Directly connected (NDC $= 4$):
area+fit, area+resize, move+fit, resize+fit

Indirectly connected (NIC $= 2$):
area+move, move+resize

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

"If a class is designed in ad hoc manner and unrelated components are included in the class, the class represents more than one concept and does not model an entity. 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*, Proceedings of the Symposium on Software Reusability (SSR), 1995

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

— Steven McConnell, *Code Complete*, 2004

Abstraction

- Color: red

- Weight: 120g

- Price: $0.99

```
1  var file = {
2      path: '/tmp/data.txt',
3      read: function() { ... },
4      write: function(txt) { ... }
5  }
```

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

Inheritance vs. Cohesion

Figure 3: Number of descendents and Class Cohesion

"Our results show that the classes that are heavily reused via inheritance exhibit lower cohesion. We expected to find that the most reused classes would be the most cohesive ones." — James M. Bieman and Byung-Kyoo Kang

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 inherit its parts."

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

Source: Inheritance Is a Procedural Technique for Code Reuse (2016)

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

Read this:

Cohesion and Reuse in an Object-Oriented System, James M. Bieman and Byung-Kyoo Kang, Proceedings of the Symposium on Software Reusability (SSR), 1995

Code Complete, Steven McConnell, 2004

Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, Edward Yourdon and Larry Constantine, Prentice Hall, 1979

Inheritance Is a Procedural Technique for Code Reuse (2016)