Object-Oriented Software Metrics

Radu Marinescu

LOOSE Research Group
Politehnica University of Timișoara
radu.marinescu@cs.upt.ro

Outline
**What are Metrics?**

- Functions, that assign a precise numerical value
- to
  - Products (Software),
  - Resources (Staff, Tools, Hardware) or
  - Processes (Software development).

**Object-Oriented Product Metrics**

- Size & Structural Complexity
- Inheritance
- Coupling
- Cohesion
Weighted Method Count (WMC)

- **Definition** [Chidamber & Kemerer, 1994]:

\[
WMC = \sum c_i
\]

where \( c_i \) = complexity of method \( m_i \)

- **Interpretation:**
  - Time and effort for maintenance
  - The higher the WMC for a class, the higher the influence on the subclasses
  - A high WMC reduces the reuse probability for the class

**GOOD:** Metric is configurable!

**BAD:** Interpretation can’t directly lead to improvement action!

Depth of Inheritance Tree (DIT)

- **Definition** [Chidamber & Kemerer, 1994]:

  depth of a class in the inheritance graph

- **Interpretation:**
  - the higher DIT, the lower the understandability of the class
  - higher DIT, class more complex
    - harder to test
  - the higher DIT, the higher the potential reuse from the superclasses

**BAD:** Nothing about real reuse!
Object-Oriented Software Metrics

Change Dependency Between Classes (CDBC)

- **Definition** [Hitz & Montazeri, 1996]:
  - number of methods in a *client-class* (CC) that depend on a *server-class* (SC)

- **Characteristics:**
  - defined on a pair of classes
  - stability of the server class
  - differentiates between types of coupling

- **Interpretation:**
  - the higher CDBC, the bigger the *maintenance* impact on CC, by a change in SC

---

CDBC differentiates between types of coupling

<table>
<thead>
<tr>
<th>Type of Relationship</th>
<th>Number of Methods Potentially Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC is not used by CC at all</td>
<td>0</td>
</tr>
<tr>
<td>SC is the class of an instance variable of CC</td>
<td>n</td>
</tr>
<tr>
<td>Local variables of type SC are used within j methods of CC</td>
<td>j</td>
</tr>
<tr>
<td>SC is a superclass of CC</td>
<td>n</td>
</tr>
<tr>
<td>SC is the parameter type of j methods of CC</td>
<td>j</td>
</tr>
<tr>
<td>CC accesses a global variable of class SC</td>
<td>n</td>
</tr>
</tbody>
</table>
Object-Oriented Software Metrics

Tight Class Cohesion (TCC)

- **Definition** [Bieman&Kang, 1995]:
  - the relative number of method-pairs that access an attribute of the class
- **Example:**
  
  ![Diagram of classes A, B, C, D, E and attributes X, Y with arrows indicating connections]

  TCC = \( \frac{2}{10} = 0.2 \)

- **Interpretation:**
  - higher TCC \( \Rightarrow \) tighter the connection between the methods
  - lower TCC \( \Rightarrow \) probably class implements more than one functionality

**GOOD:** Interpretation can lead to improvement action!

**GOOD:** Ratio values allow comparison between systems!

Obstacles in Using Metrics

- **Interpretation of metrics is hard**
  - many confusing and redundant definitions
  - issue of thresholds
    - need statistical data
  - hard to compare the results
    - normalize!

- **Applying metrics is hard**
  - issue of granularity
    - metrics need to be used in combination
      - quality models
      - detection strategies
      - polymetric views
Issue of thresholds exemplified

- Let's play a game:
  - Want brief overview of the code of an OO system never seen before
  - Want to find out how hard it will be to understand the code

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>35,000</td>
</tr>
<tr>
<td>NOM</td>
<td>3,600</td>
</tr>
<tr>
<td>NOC</td>
<td>380</td>
</tr>
</tbody>
</table>

We need statistical data for thresholds

Several questions remain unanswered...

- Is it “normal” to have...
  - ...380 classes in a system with 3,600 methods?
  - ...3,600 methods in a system with 35,000 lines of code?

- What means NORMAL?
  - i.e. how do we compare with other projects?

- What about the hierarchies? What about coupling?

1. We need means of comparison. Thus, proportions are important!
   - Collect further relevant numbers; especially coupling and use of inheritance
**Issue of thresholds**

- Interpretation based on a statistically relevant collection of data
  - collected for Java and C++
  - over 80 systems

<table>
<thead>
<tr>
<th>Metric</th>
<th>Java Average</th>
<th>High</th>
<th>C++ Average</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC/Line of Code</td>
<td>6.0 ± 0.26</td>
<td>5.35</td>
<td>6.0 ± 0.25</td>
<td>5.0</td>
</tr>
<tr>
<td>LOC/Operation</td>
<td>7.0 ± 0.35</td>
<td>5.0</td>
<td>7.0 ± 0.35</td>
<td>5.0</td>
</tr>
<tr>
<td>NOM/Class</td>
<td>4.0 ± 0.35</td>
<td>5.0</td>
<td>4.0 ± 0.35</td>
<td>5.0</td>
</tr>
<tr>
<td>NOM/Package</td>
<td>1.0 ± 0.35</td>
<td>5.0</td>
<td>1.0 ± 0.35</td>
<td>5.0</td>
</tr>
<tr>
<td>CALLS/Operation</td>
<td>2.0 ± 0.35</td>
<td>1.15</td>
<td>2.0 ± 0.35</td>
<td>1.15</td>
</tr>
<tr>
<td>FANDT/Ocall</td>
<td>0.5 ± 0.35</td>
<td>0.35</td>
<td>0.5 ± 0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>ANDC</td>
<td>0.25 ± 0.35</td>
<td>0.19</td>
<td>0.25 ± 0.35</td>
<td>0.19</td>
</tr>
<tr>
<td>AHH</td>
<td>0.00 ± 0.35</td>
<td>0.00</td>
<td>0.00 ± 0.35</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Overview Pyramid**

[Lanza, Marinescu 2006]

---

**The need to aggregate metrics**

- Quality Models to correlate quality criteria with concrete measures

**Factor-Criteria-Metrics model**

[J.A. McCall, 1977]
Metrics are too fine-grained indicators

What do we expect from quality models?
- **diagnosis**, **location** and **treatment hints** for quality problems

FCM **locates** classes and methods with abnormal metric values

In FCM models it is hard to find the proper treatment for quality problems because abnormal metric values are rather symptoms than causes of poor quality.

Metrics should be used in a goal-oriented fashion

**Goal-Question-Metric Approach**  
[Basili & Rombach, 1988]

- **Define a Goal**
  - How efficient is the ACME tool

- **Formulate Questions**
  - Who uses the ACME tool?
  - How high is productivity/quality with/without the ACME tool?

- **Find suitable Metrics**
  - Percent of developers that use the ACME tool
  - Experience with ACME
  - Size, complexity, solidity, ... of code
Goal-oriented aggregation of metrics

Detection Strategies
[Marinescu 2002, 2004]

Metrics-based rules that capture quality aspects
- based on filtering and composition

Filtering
Return a subset of data based on thresholds

Composition
Compose metrics in a high level rule

Combine Metrics in a Visual Manner

Polymetric Views
[Lanza, Ducasse 2003]

Use simple metrics and layout algorithms.

Visualize up to 5 metrics per node
A Picture is Worth a Thousand Words...

System Complexity View of ArgoUML

System Hotspots

- width = height = NOM
- gray-level = LOC
Quickly “Reading” Classes

- Visualization Technique
  - serves as code inspection technique
  - reduces the amount of code that must be read

Class Blueprint
[Lanza, Ducasse 2001]

The Class Blueprint
Books on object-oriented metrics

Object-Oriented Software Metrics

Instead of conclusions....

- **What metrics** do we use?
  - It depends…on our **measurement goals**

- **What information** to retrieve?
  - It depends…on our **objectives**

- **What entities** do we measure?
  - It depends…on the **language**

  Can we understand the beauty of a painting by…
  …measuring its frame or counting the colors?

**DISCLAIMER:**
Metrics are not enough to understand and evaluate design!