Object Oriented Metrics

Ali Kamandi
Sharif University of Technology
kamandi@ce.sharif.edu

Spring 2007
OO Metrics: Introduction

- Measurement and metrics are key components of any engineering discipline.
- The use of metrics for OO systems has progresses much more slowly than the use of OO methods.
- As OO systems become more pervasive, it is essential that software engineers have quantitative measurements for assessing the quality of designs at both the architectural and component levels.

The intent of OO (or non-OO) metrics:
- To better understand the quality of the product.
- To assess the effectiveness of the process.
- To improve the quality of work performed at a project level.

[Pressman 2001]
Principles of Metrics

- Metrics must be
  - Simple
  - Easy to collect (automated)
  - Easy to interpret

- The selection of an extensive set of metrics will depend on the project's characteristics and context
Metrics for the OO Design Model

- Whitmire[WHI97] describes several distinct and measurable characteristics of an OO design:
  - **Size** (Population, Volume, Length, Functionality)
  - Complexity
  - Coupling/Cohesion
  - Completeness
  - **Similarity** (the degree to which two or more classes are similar)
  - **Volatility** (the degree of change in an artifact because of defects or changing requirements)
Software Metrics

- **Some Traditional Metrics**
  - McCabe Cyclomatic Complexity (CC)
  - Lines of Code (LOC)
  - Comment Percentage (CP)

- **Some Object-Oriented Metrics**
  - Weighted Methods Per Class (WMC)
  - Coupling Between Object Classes (CBO)
  - Response for a Class (RFC)
  - Depth of Inheritance Tree (DIT)
  - Lack of Cohesion in Methods (LOCM)
Researches on OO Metrics

- Chidamber and Kemerer: CK Metrics Suite, 1994
- CK suite validated by Basili in 1996 and again by Tang in 1999
- Many other object-oriented metrics are derived from the CK suite of object-oriented metrics
- Lorenz and Kidd 1994
- Harrison, Counsell and Nithi, MOOD Metric Suite, 1998
- Whitmire: OO Design Measurements, 1997
- ...

...
Class-Oriented Metrics 1

- Chidamber and Kemerer 1994:
  - **WMC: weighted methods per class** ($\Sigma$ complexity of methods) (low)
    - Amount of effort required to implement and test a class
  - The larger the number of methods, the more complex is the inheritance tree.
  - Large number of methods: Application specific and limiting potential reuse
Class-Oriented Metrics 2

- Chidamber and Kemerer 1994:
  
  • **DIT: depth of the inheritance tree** (Max length node to root) (low)
    
    - Lower level classes inherit many methods.
    - Difficulties when attempting to predict the behavior of a class.
    
    - Greater design complexity.
    - Positive: many methods may be reused
Class-Oriented Metrics 3

- Chidamber and Kemerer 1994:
  - **NOC: number of children** (subclasses that are immediately subordinate to a class)
    - High: reuse, the amount of test
Chidamber and Kemerer (Continue):

- **CBO:** coupling between object classes (No. of coll. in CRC) (low)
  - High: reusability will decrease, complicated testing and modifications
  - This is consistent with the general guideline to reduce coupling.
Class-Oriented Metrics 5

- Chidamber and Kemerer (Continue):
  - **RFC: responsibility for a class** (No. of methods in response set) (low)
    - High: Testing and overall design complexity will increase.
  - **LCOM: lack of cohesion in methods** (No of methods that access one or more of the same attributes) (low)
    - High: Complexity of class design, class might be better designed by breaking it into two or more separate classes.
    - It is desirable to keep cohesion high; that is keep LCOM low.
Class-Oriented Metrics 6

- Lorenz and Kidd [LOR94]:
  - **Size-oriented**: Counts of attributes and operations.
  - **Inheritance-based**: The manner in which operations are reused through the class hierarchy.
  - **Internals**: Look at cohesion and code-oriented issues.
  - **Externals**: Examine coupling and reuse.

- A Sampling of Metrics:
  - **CS**: class size (no. of operations) (no. of attributes) (low)
    - High: class has too much responsibility, reduce the reusability, complicate implementation and testing
  - **NOO**: number of operations overridden by a subclass (low)
  - **NOA**: number of operations added by a subclass
  - **SI**: specialization index = NOO * level / total no of methods
Class-Oriented Metrics

The MOOD Metrics Suite: Harrison, Counsell and Nithi, 1998

- Method inheritance factor: inherited methods / total methods
- Coupling factor: formula
- Polymorphism factor: formula
Operation-Oriented Metrics

Lorenz and Kidd [LOR94]:

- average operation size
- operation complexity
- average number of parameters per operation
Testability Metrics

Proposed by Binder [BIN94]:

- encapsulation related
  - lack of cohesion in methods
  - percent public and protected
  - public access to data members

- inheritance related
  - number of root classes
  - fan in: multiple inheritance
  - number of children and depth of inheritance tree
OO Project Metrics

Proposed by Lorenz and Kidd [LOR94]:

- number of scenario scripts
- number of key classes
- number of subsystems
## Req. and Use Case Model Metrics [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
</table>
| **Size**       | Number of requirements  
                 | Number of Use Cases  
                 | Number of scenarios, total and per use case  
                 | Number of actors  
                 | Length of Use Case (pages of event flow, for example) |
| **Effort**     | Staff-time units (with production, change and repair separated) |
| **Volatility** | Number of defects and change requests |
| **Quality**    | Defects - number of defects, by severity  
                 | Reported complexity (0-5, by analogy with COCOMO [BOE81]) |
## Requirements and Use Case Model Metrics (2)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completeness</td>
<td>Use Cases completed (reviewed and under configuration management with no defects outstanding)/use cases identified (or estimated number of use cases)</td>
</tr>
</tbody>
</table>
| Traceability    | Analysis  
  Scenarios realized in analysis model/total scenarios  
  Design  
  Scenarios realized in design model/total scenarios  
  Implementation  
  Scenarios realized in implementation model/total scenarios  
  Test  
  Scenarios realized in test model (test cases)/total scenarios  
  Requirements-to-UC Traceability = Traceable to Use-Case Model/Total number of requirements |
## OO Design Model Metrics [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
</table>
| **Size**       | Number of classes  
                  Number of subsystems  
                  Number of subsystems of subsystems …  
                  Number of packages  
                  Methods per class, internal, external  
                  Attributes per class, internal, external  
                  Depth of inheritance tree  
                  Number of children |
| **Effort**     | Staff-time units (with production, change and repair separated) |
| **Volatility** | Number of defects and change requests (open, closed) |
## OO Design Model Metrics (2) [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality</td>
<td>Complexity: Response For a Class (RFC): this may be difficult to calculate because a complete set of interaction diagrams is needed.</td>
</tr>
<tr>
<td></td>
<td>Coupling: Number of children, Coupling between objects (class fan-out)</td>
</tr>
<tr>
<td></td>
<td>Cohesion: Number of children</td>
</tr>
<tr>
<td></td>
<td>Defects: Number of defects, by severity (open, closed)</td>
</tr>
<tr>
<td>Completeness</td>
<td>Number of classes completed/number of classes estimated (identified)</td>
</tr>
<tr>
<td>Traceability</td>
<td>Number of classes in Implementation Model/number of classes</td>
</tr>
</tbody>
</table>
### OO Implementation Metrics (1) [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
</table>
| **Size**       | Number of classes  
                | Number of components  
                | Number of implementation subsystems  
                | Number of subsystems of subsystems …  
                | Number of packages  
                | Methods per class, internal, external  
                | Attributes per class, internal, external  
                | Size of methods, Size of attributes  
                | Depth of inheritance tree  
                | Number of children  
                | Estimated size at completion |
| **Effort**     | Staff-time units (with production, change and repair separated) |
| **Volutility** | Number of defects and change requests (open, closed)  
                | Breakage for each corrective or perfective change, estimated (prior to fix) and actual (upon closure) |
## OO Implementation Metrics (2) [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
</table>
| **Quality**    | Complexity | Response For a Class (RFC)  
|                |            | Cyclomatic complexity of methods* |
| **Coupling**   | Number of children  
|                |            | Coupling between objects (class fan-out)  
|                |            | Message passing coupling (MPC) [Li and Henry 1993] |
| **Cohesion**   | Number of children  
|                |            | Lack of cohesion in methods (LCOM) |
| **Defects**    | Number of defects, by severity, open, closed |
| **Completeness** | Number of classes unit tested/number of classes in design model  
|                |            | Number of classes integrated/number of classes in design model  
|                |            | Active integration and system test time (accumulated from test process), that is, time with system operating (used for maturity calculation) |
## Test Metrics [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Number of Test Cases, Test Procedures, Test Scripts</td>
</tr>
<tr>
<td>Effort</td>
<td>Staff-time units for production of test cases, and so on</td>
</tr>
<tr>
<td>Volatility</td>
<td>Number of defects and change requests (open, closed)</td>
</tr>
<tr>
<td>Quality</td>
<td>Defects — number of defects by severity, open, closed (these are defects raised against the test model itself)</td>
</tr>
<tr>
<td>Completeness</td>
<td>Number of test cases written/number of test cases estimated</td>
</tr>
<tr>
<td></td>
<td>Code coverage</td>
</tr>
<tr>
<td>Traceability</td>
<td>Number of successful Test Cases in Test Evaluation Summary/Number of test cases</td>
</tr>
</tbody>
</table>
## Change Management Metrics [RUP]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Number of defects, change requests by severity and status, also categorized as number of perfective changes, number of adaptive changes and number of corrective changes.</td>
</tr>
<tr>
<td>Effort</td>
<td>Defect repair effort, change implementation effort in staff-time units</td>
</tr>
<tr>
<td>Volatility</td>
<td>Breakage (estimated, actual) for the implementation model subset.</td>
</tr>
<tr>
<td>Completeness</td>
<td>Number of defects discovered/number of defects predicted</td>
</tr>
</tbody>
</table>
Process Metrics

- Duration
- Effort
- Training time
- Inspection time
- Meeting time
- Process Problem
Project Metrics

- **Cost/Schedule Control**
  - BCWS, Budgeted Cost for Work Scheduled
  - BCWP, Budgeted Cost for Work Performed
  - ACWP, Actual Cost of Work Performed

- **Modularity** = average breakage (NCNB*) per perfective or corrective change on implementation model

- **Adaptability** = average effort per perfective or corrective change on implementation model

- ...

- * NCNB is non-comment, non-blank code size.
Data Analysis: Example (1)

- **Projects**
  - Project A: 46 Java Classes (Commercial Software)
  - Object-Oriented Constructs
  
  - Project B: 1000 Java Classes (NASA Software)
  - Excellent Object-Oriented Constructs
  
  - Project C: 1617 C++ Classes (NASA Software)
  - Good Object-Oriented Constructs
Data Analysis: Example (2)

- Correlation of Metrics over Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>CBOxRFC</th>
<th>CBOxWMC</th>
<th>RFCxWMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project A</td>
<td>0.83</td>
<td>0.43</td>
<td>0.47</td>
</tr>
<tr>
<td>Project B</td>
<td>0.28</td>
<td>0.11</td>
<td>0.75</td>
</tr>
<tr>
<td>Project C</td>
<td>0.35</td>
<td>0.26</td>
<td>0.83</td>
</tr>
</tbody>
</table>

- Metric 1 x Metric 2 = correlation between metric 1 and metric 2.
Data Analysis: Example (3)

- **Regression Model**
  - $RFC = \beta_1 WMC + \beta_2 CBO + \text{Constant}$

<table>
<thead>
<tr>
<th></th>
<th>$\beta_{WMC}$</th>
<th>$\beta_{CBO}$</th>
<th>Constants</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC$_A$</td>
<td>0.131</td>
<td>0.777</td>
<td>8.036</td>
<td>0.70</td>
</tr>
<tr>
<td>RFC$_B$</td>
<td>0.732</td>
<td>0.200</td>
<td>15.427</td>
<td>0.60</td>
</tr>
<tr>
<td>RFC$_C$</td>
<td>0.792</td>
<td>0.148</td>
<td>6.039</td>
<td>0.70</td>
</tr>
</tbody>
</table>
References

Questions?