Object Oriented Metrics

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

November 2006

OO Metrics: Introduction

[Pressman]

- 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]

[RUP]

Principles of Metrics

- Metrics must be
  - Simple
  - Objective
  - Easy to collect (automated)
  - Easy to interpret
  - Hard to misinterpret
- The selection of an extensive set of metrics will depend on the project's characteristics and context

[Pressman]

Metrics for the OO Design Model

- Whitmire[WHI97] describes nine distinct and measurable characteristics of an OO design:
  - **Size** (Population, Volume, Length, Functionality)
  - **Complexity** (Interrelated classes)
  - **Coupling** (No. of Collaborations between classes or messages)
  - **Sufficiency** (the degree to which a design component possesses features in its abstraction from the current application point of view)
  - **Completeness** (same as sufficiency but consider multiple points of view for fully representation of problem domain)
  - **Cohesion** (the degree to which the set of properties it possesses is part of the problem or design domain)
  - **Primitiveness** (the degree to which an operation is atomic)
  - **Similarity** (the degree to which two or more classes are similar)
  - **Volatility** (the degree of change or churn 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
- …

[Pressman]

Class-Oriented Metrics 1

- Chidamber and Kemerer 1994:
  - **WMC: weighted methods per class** (Σ complexity of methods) (low)
    - Amount of effort required to implement and test a class
    - Complexity of inheritance tree.
    - Application specific and limiting potential reuse
  - **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
  - **NOC: number of children** (subclasses that are immediately subordinate to a class)
    - High: reuse, the amount of test

[Pressman]

Class-Oriented Metrics 2

- 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 guidline to reduce coupling.
  - **RFC: response 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.

[Pressman]

Class-Oriented Metrics 3

- 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

[Pressman]

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

[Pressman]

Operation-Oriented Metrics

Lorenz and Kidd [LOR94]:

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

[Pressman]

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

[Pressman]

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]

Characteristic	Metrics
Size	Number of requirements Number of Use Cases Number of Use Case Packages 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 (open, closed)
Quality	Defects - number of defects, by severity (open, closed) Reported complexity (0-5, by analogy with COCOMO [BOE81])

Requirements and Use Case Model Metrics (2)

Characteristic	Metrics
Completeness	Use Cases completed (reviewed and under configuration management with no defects outstanding)/use cases identified (or estimated number of use cases)
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]

Characteristic	Metrics
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]

Characteristic	Metrics	
Quality	Complexity	Response For a Class (RFC): this may be difficult to calculate because a complete set of interaction diagrams is needed.
	Coupling	Number of children Coupling between objects (class fan-out)
	Cohesion	Number of children
	Defects	Number of defects, by severity (open, closed)
Completeness	Number of classes completed/number of classes estimated (identified)	
Traceability	Number of classes in Implementation Model/number of classes	

OO Implementation Metrics (1) [RUP]

Characteristic	Metrics
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)
Volatility	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]

Characteristic	Metrics	
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]

Characteristic	Metrics
Size	Number of Test Cases, Test Procedures, Test Scripts
Effort	Staff-time units for production of test cases, and so on
Volatility	Number of defects and change requests (open, closed)
Quality	Defects — number of defects by severity, open, closed (these are defects raised against the test model itself)
Completeness	Number of test cases written/number of test cases estimated Code coverage
Traceability	Number of successful Test Cases in Test Evaluation Summary/Number of test cases

Change Management Metrics [RUP]

Characteristic	Metrics
Size	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.
Effort	Defect repair effort, change implementation effort in staff-time units
Volatility	Breakage (estimated, actual) for the implementation model subset.
Completeness	Number of defects discovered/number of defects predicted

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

	CBOxRFC	CBOxWMC	RFCxWMC
Project A	0.83	0.43	0.47
Project B	0.28	0.11	0.75
Project C	0.35	0.26	0.83

- Metric 1 x Metric 2 = correlation between metric 1 and metric 2.

Data Analysis: Example (3)

- **Regression Model**
  - - RFC = β_1WMC + β_2CBO + Constant

	β_{WMC}	β_{CBO}	Constants	R^2
RFC_A	0.131	0.777	8.036	0.708
RFC_B	0.732	0.200	15.427	0.608
RFC_C	0.792	0.148	6.039	0.709

References

- [PRE01] Pressman, R. S., Software Engineering, A Practitioners Approach, 5th Ed., McGraw-Hill, 2001.
- [ROY98] Walker Royce 1998. *Software Project Management: A Unified Framework.* Addison Wesley Longman.
- [CHI94] Chidamber, S. R. and C. F. Kemerer, “A Metrics Suite for Object Oriented Design”, IEEE Trans. Software Engineering, vol. SE-20, no. 6, June 1994, pp. 476-493.
- [HAR98] Harrison, R. S.J. Counsell, and R. V. Nithi, “An Evaluation of the MOOD set of Object-Oriented Software Metrics,” IEEE Trans. Software Engineering, vol. SE-24, no. 6, June 1998, pp. 491-496.
- [LOR94] Lorenz, M. and J. Kidd, Object-Oriented Design Metrics, Prentice-Hall, 1994.
- [WHI97] Whitmire, S., Object-Oriented Design Measurement, Wiley, 1997.
- [BIN94] Binder, R. V., “Object-Oriented Software Testing,” CACM, vol. 37, no. 9, september 1994, p. 29.

Questions?