Object Oriented Software Metrics

Jana Polgar
Monash University, Australia

INTRODUCTION

Software measurement is considered to be an efficient means to monitor the quality of software projects, predict cost of maintenance, assess reusability of components, provide prediction of faults in similar projects, and contribute to improvement of the software development process. This chapter surveys software metrics literature with particular focus on object-oriented metrics, and metrics for measuring code complexity. Furthermore, we provide a critical view of software metrics and their usability.

BRIEF SURVEY OF OBJECT-ORIENTED METRICS

Since 1995, the trend toward incorporating *measurement theory* into all software metrics has led to identification of scales for measures, thus providing some perspective on dimensions. The most common scale types based on measurement theory are as follows:

- *Ordinal scale*: An ordered set of categories (often used for adjustment factors in cost models based on a fixed set of scale points)
- *Interval scale*: Numerical values, where the difference between each consecutive pair of numbers is an equivalent amount, but there is no "real" zero value
- *Ratio scale*: Elements are assigned numbers such that differences and ratios between the numbers reflect differences and ratios of the attribute
- *Nominal scale*: A set of categories into which an item is classified.
- *Absolute scale*: Elements are assigned numbers such that all properties of the numbers reflect analogous properties of the attribute

Fetchke (1995) and Zuse (1994) analyzed the properties of object-oriented software metrics on the basis of measurement theory. The underlying notion of measurement theory is based on intuitive or empirical existence of relationships among objects within our Universe of Discourse. These relationships can then be formally described in a mathematically derived formal relational system. They also investigated how and under what conditions the software measures may be viewed as ordinal, ratio, nominal, and interval. They admitted that these scale types present very little meaning with regard to maintainability and "error-proneness" of the application.

The contribution of Zuse and Fetchke's work is in the introduction of specific perspectives of measures. They emphasize preciseness of definition of scales as well as definition of an attribute that is measured.

The *axiomatic approach* was proposed by Weyuker (1988). This framework is based on a set of nine axioms, as listed in Table 1.

In Weyeker's metric proposal, we observe the formalization of structural inheritance complexity metrics. Property 9 means that splitting one class into two classes can reduce the complexity. The experience supports argument by Chidamber and Kemerer (1994) that the complexity of interaction may even increase when classes are divided.

Fenton and Pfleger (1997) used the term "software metrics" to describe the following artifacts:

- A number derived, usually empirically, from a process or code [for example, lines of code (LOC) or number of function points]
- A scale of measurement (The example used in Fenton's book is nominal scale or classification.)
- An identifiable attribute used to provide specific functionality (an example is "portability" or class coupling metric)
- Theoretical or data-driven model describing a dependent variable as a function of independent variables (an example can be the functional relationship between maintenance effort and program size)

These descriptions typically lead to widespread confusion between models and their ability to predict desired software characteristics, thus their suitability in being used for estimation purposes.

The metrics of Chidamber and Kemerer, summarized in Table 2, also have foundation in measurement theory. The authors do not base their investigation on the extensive structure. The criticism by Churcher and Sheppard (1994) points to the ambiguity of some metrics, particularly WMC. Hitz and Montazeri (1996) and Fetchke (1995) showed that CBO does not use a sound empirical relation

Table 1. Weyuker's axioms

Axiom	Name	Description
1	Noncoarseness	$(\exists P)(\exists Q)(\mu(P) \neq \mu(QQ))$
2	Granularity	Let c be non-negative number. Then there is finite number of class with the complexity = c
3	Nonuniquesniss	There is distinct number of classes P and Q such that $\mu(P)=\mu(Q)$
4	Design detail matter	$(\exists P)(\exists Q)(P \equiv Q and \mu(P) \neq \mu(Q))$
5	Monotonicity	$(\forall P)(\forall Q)(\mu(P) \leq \mu(P+Q) and \mu(Q) \leq \mu(P+Q))$
6	Non-equivalence of interaction	a) $(\exists P)(\exists Q)(\exists R)\mu(P)=\mu(Q) and \mu(P+R) \neq \mu(Q+R)$ b) $(\exists P)(\exists Q)(\exists R)\mu(P)=\mu(Q) and \mu(R+P) \neq \mu(R+Q)$
7	Interaction among statements	Not considered among objects
8	No change on renaming	If P is renaming of Q then $\mu(P)=\mu(Q)$
9	Interaction CAN increase complexity	$(\exists P)(\exists Q)(\mu(P)+\mu(Q)<\mu(P+Q))$

Table 2. Chidamber and Kemerer metrics (Chidamber, 1994)

Weighted Methods per Class **(WMC)**	$WMC=\sum_{i=1}^{n} c_i$ where c_i is the static complexity of each of the n methods
Depth of Inheritance Tree **(DIT)**	With multiple inheritance the max DIT is the length from node to the root
Number of Children **(NOC)**	Number of immediate subclasses
Coupling Between Object Classes **(CBO)**	Number of other classes to which a particular class is coupled. CBO maps the concept of coupling for a class into a measure.
The Response for a Class **(RFC)**	The size of response set for a particular class.
The lack of Cohesion metric **(LCOM)**	$LCOM=\lvert P\rvert-\lvert Q\rvert if \lvert P\rvert>\lvert Q\rvert=0 otherwise$

system, particularly, that it is not based on the extensive structures. Furthermore, LCOM metric allows representation of equivalent cases differently, thus introducing additional error.

Coupling measures form an important group of measures in the assessment of dynamic aspects of design quality. Coupling among objects is loosely defined as the measure of the strength of the connection from one object to another. The approaches of different authors mostly differ in definition of the measured attribute—coupling among classes. Table 3 provides a summary of differences in definitions. Some of the attributes may be known only too late in development.

Two aspects impact coupling between classes: the frequency of messaging between classes (cardinality and multiplicity of objects derived from these classes) and type of coupling. The discussion in Eder and Kappel (1994) distinguishes among three types: interaction coupling, component coupling, and inheritance coupling. The degree of coupling is based on defining a partial order on the set of coupling types. The low end is described by small and explicit interrelationships, and the high end of the scale is assigned to large, complex and implicit interrelationships. The definition is subjective and requires empirical assignment of values in order to be used as a software quality indicator.

Cohesion is defined as a degree to which elements in a class belong together. The desirable property of a good design is to produce a highly cohesive classes. Comparison of different frameworks and thorough discussion can be found in Briand's work (Briand, Daly, & Wurst, 1997). Eder (Eder & Kappel, 1994) provided a comprehensive