Advances in Computer, Signals and Systems (2016) 1: 28-32
Clausius Scientific Press, Canada

Formalizing UML Model Metrics Using Z Language

Fangjun Wu[1, 2,a]

[1]School of Information Technology, Jiangxi University of Finance and Economics, Nanchang, China
[2]Jiangxi Key Laboratory of Data and Knowledge Engineering, Jiangxi University of Finance and Economics, Nanchang, China

[a]wufangjun@jxufe.edu.cn

Keywords: software measurement, object-oriented, UML class diagrams, Z language, empirical validation, theoretical verification

Abstract: Till now, a large variety of researchers have carried out lots of efforts on object-oriented and UML model metrics from different views. They put forward numerous of metrics and carried out some series of theoretical and experimental verifications on understandability, analyzability, maintainability, fault-proneness, change-proneness and reuse. However, there is no formal semantic specification for UML model metrics, which may lead to potential semantic inconsistency and ambiguity. To solve this problem, this paper provided formalization for UML model metrics at the level of UML Meta models. This formalization can not only help people to understand the meaning of UML model metrics, but also can be used in the application domain of UML model metrics in a more rigorous way.

1. Introduction

Till now, a large variety of researchers have carried out lots of efforts on object-oriented and UML model metrics from different views. They put forward variety of metrics [1-4]. At the same time, researchers carried out some series of theoretical and experimental verifications[5,6], especially predicted external characteristics of software based on object-oriented and UML model metrics, such as understandability], analyzability, maintainability, fault-proneness, change-proneness and reuse.

However, there is a small amount of formalization for object-oriented metrics [7-10], meanwhile there is little formalization for UML model metrics, which may lead to different interpretations, potential semantic inconsistency and ambiguity. To solve this problem, this paper provides a formal description for UML model metrics. This formalization can not only help people to understand the meaning of object-oriented metrics, but also can be used in the application domain of object-oriented metrics in a more rigorous way.

2. Related Work

Software measurement is a long well known issue and has garnered much attention. Till now, numerous metrics have been proposed from structural measurement, such as Halstead's metrics, McCabe's cyclomatic number, to object-oriented measurement, such as CK suite [1], MOOD suite[2], QMOOD suite[3], and then to UML model measurement, such as Genero's suite[4].

Although lots of efforts have been developed on object-oriented and UML model metrics, there is a small amount of formalization for object-oriented metrics [7-10]. Ref.[7] formalized Depth of Inheritance Tree (abbreviated as *DIT*) and External Inheritance Factor (abbreviated as *EIF*) using the Object Constraint Language (abbreviated as OCL) based on UML Meta models. Similarly, Ref.[8] formalized Operations Hiding Factor (abbreviated as *OHF*) and Attribute Inheritance Factor (abbreviated as *AIF*) using OCL based on UML meta models. Formal definitions of the MOOSE suite were given in Ref.[9] using OCL. A suite of reusability metrics, such as Rate of Component Customizability (abbreviated as *RCC*), were formalized in Ref.[10] using OCL. By contrast with a small amount of formalization for object-oriented metrics, there is little formalization for UML model metrics.

3. Formalizing UML Model Metrics

On the basis of formalization of some core constructs of UML models, such as attributes, parameters, methods and classes done by Soon-Kyeong Kim and David Carrington[11], this paper formalized Genero's UML model suite using Z schemas [12] at the level of UML meta models. For unity, the same symbols were used in this paper. Given a UML class diagram *D*, a class *c*,

- Free set [*Name*] was used to represent names of classes, attributes and methods.
- Free type *Type* was used to describe data type of attributes and parameters of methods.
- Free type *VisibilityKind* was used to describe private, public, protected characteristics of attributes and methods.
- Free set [*Expression*] was used to set initial values of attributes and parameters of methods.
- Free type *RelationshipKind* was used to describe all kinds of relationships between classes.
- Free type *NavigabilityKind* was used to describe the navigation direction of the relationship between classes.

UMLClassDiagram

$name: Name$

$class: \mathbb{F}\ Class$

$kind: RelationshipKind$

$multiplicity: \mathbb{P}_1 \mathbb{N}$

$navigability: NavigabilityKind$

$\forall c_1, c_2: class \bullet c_1.name = c_2.name \Rightarrow c_1 = c_2$

$kind = dependence \Rightarrow$

$\quad (\exists c_1, c_2: class \bullet$

$\quad\quad (c_1 \neq c_2 \wedge \text{there is a dependence relationship from } c_1 \text{ to } c_2$

$\quad\quad \wedge (\exists n: navigability \bullet n = one)))$

kind= association $\Rightarrow$

$(\exists c_1,c_2{:}class \bullet c_1 \neq c_2 \wedge$

there is a association relationship between c_1 *and* $c_2 \wedge$

$multiplicity \neq \{0\} \wedge (\exists n{:}navigability \bullet n=none))$

kind= association class $\Rightarrow$

$(\exists c_1,c_2,c_3{:}class \bullet c_1 \neq c_2 \neq c_3 \wedge$

there is a association class c_3 *between* c_1 *and* $c_2 \wedge$

$multiplicity \neq \{0\} \wedge (\exists n{:}navigability \bullet n=none))$

kind= aggregation $\Rightarrow$

$(\exists c_1,c_2{:}class \bullet c_1 \neq c_2 \wedge$

there is a aggregation relationship from c_1 *to* $c_2 \wedge$

$multiplicity \neq \{0\} \wedge (\exists n{:}navigability \bullet n=one))$

kind= combination $\Rightarrow$

$(\exists c_1,c_2{:}class \bullet c_1 \neq c_2 \wedge$

there is a combination relationship from c_1 *to* $c_2 \wedge$

$multiplicity \neq \{\{0,1\},\{1\}\} \wedge (\exists n{:}navigability \bullet n=one))$

kind= generalization$\Rightarrow$

$(\exists c_1,c_2{:}class \bullet c_1 \neq c_2 \wedge$

there is a generalization relationship from c_1 *to* $c_2 \wedge$

$\{ c_1,c_2\}^* \cap id(\downarrow Class)=\varnothing \wedge (\exists n{:}navigability \bullet n=one))$

NCMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

class: $\mathbb{F}$ *Class*

NC: $\mathbb{P}_1\mathbb{N}$

$NC=\#\{c_1 | \forall c_1{:}class \}$

NAMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

class: $\mathbb{F}$ *Class*

attribute: $\mathbb{F}$ *Attribute*

NA: $\mathbb{P}_1\mathbb{N}$

$NA=\#\{a_1 | \forall a_1{:}\ attribute \bullet \exists c_1{:}class\}$

NMMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

class: $\mathbb{F}$ *Class*

method: $\mathbb{F}$ *Method*

NM: $\mathbb{P}_1\mathbb{N}$

$NM=\#\{m_1 | \forall m_1{:}method \bullet \exists c_1{:}class\}$

NAssocVCMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

NAssocVC: $\mathbb{P}_1\mathbb{R}$

NAssocVC= NAssoc/NC

NAggVCMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

NAggVC: $\mathbb{P}_1\mathbb{R}$

NAggVC= NAgg/NC

NDepVCMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

NDepVC: $\mathbb{P}_1\mathbb{R}$

NDepVC=NDep/NC

NGenMetric

classdiagram: $\mathbb{F}$ *UMLClassDiagram*

class: $\mathbb{F}$ *Class*

kind: RelationshipKind

NGen: $\mathbb{P}_1\mathbb{N}$

$NGen = \#\{<c_1,c_2> \mid \forall c_1,c_2{:}class \bullet c_1 \neq c_2 \wedge$

$kind=generalization\ \}$

4. Summary

On the basis of formalization of some core constructs of UML models done by Soon-Kyeong Kim and David Carrington, this paper formalized Genero's UML model suite using Z schemas at the level of UML Meta models, which laid the foundation for the theoretical verification. In the next step, the nine properties of E. Weyuker and the theories of L. C. Briand will be verified to check whether they are good metrics or not and lay a theoretical foundation for their applications.

Acknowledgements

This work was financially supported by the Natural Science Foundation of China (Project No. 61163007) and Natural Science Foundation of Jiangxi (Project No. 20142BAB207010).

References

[1] S. Chidamber and C. Kemerer. A Metrics Suite for Object-oriented Design. IEEE Transactions on Software Engineering, Vol. 20, (1994), p. 476-493.

[2] F. B. Abreu. MOOD-metrics for object-oriented design. in: Proceedings of the 9th Annual Conference on Object-Oriented Programming Systems, Languages and Applications (1994) October, New York.

[3] Jagdish Bansiya and Carl G. Davis. A hierarchical model for object-oriented design quality assement. IEEE Trans on Software Engineering, Vol. 28, no. 1, (2002), p. 4-17.

[4] M. Genero. Defining and Validating Metrics for Conceptual Models (PhD thesis), University of Castilla-La Mancha, Ciudad Real, (2002).

[5] Fangjun Wu. Which One is Better, Simple or Complex Metrics. Journal of Computer and Communications, Vol. 3, no. 11, (2015), p. 52-57.

[6] Fangjun Wu. Comparative Empirical Analysis of Software Network and CK Metrics: Implications for Pre- and Post-release Faults. Journal of Software, Vol. 9, no. 3, (2014), p. 541-552.

[7] F. B. Abreu. Using OCL to formalize object-oriented metrics definitions. Technical Report ES007/2001, INESC, Portugal, http://ctp.di.fct.unl.pt/QUASAR/, May 2001.

[8] A. L. Baroni and F. B. Abreu. Formalizing Object-Oriented Design Metrics upon the UML Meta-Model. in: Proceedings of the Brazilian Symposium on Software Engineering (2002), Gramado-RS, Brazil.

[9] Aline Lucia Baroni and F. B. Abreu. An OCL-based formalization of the MOOSE metric suite. in: Proceedings of the 7th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (2003) July, Darmstadt, Germany.

[10] M. Goulão and F. B. Abreu. Formalizing metrics for COTS. in: Proceedings of ICSE Workshop on Models and Processes for the Evaluation of COTS Components (2004) May, Edinburgh, Scotland.

[11] Soon-Kyeong Kim and David Carrington. A Formal Mapping between UML Models and Object-Z Specifications. in: Proceedings of the First International Conference of Formal Specification and Development in Z and B (2000) August, York, UK.

[12] J. M. Spivey: *The Z notation: a reference manual (2nd edition)* (Prentice Hall, London, 1992).