

文章编号:1007-5321(2018)01-0109-06

DOI:10.13190/j.jbupt.2017-143

# 面向对象软件耦合度量方法

马 健<sup>1</sup>, 刘 峰<sup>1</sup>, 樊建平<sup>2</sup>

(1. 北京交通大学 计算机与信息技术学院, 北京 100044; 2. 中国科学院 深圳先进技术研究院, 深圳 518055)

**摘要:** 针对面向对象设计的 C&K 度量组中耦合度量存在的问题, 提出了一组分解的面向对象软件耦合度量方法. 参考统一建模语言类图的定义分析了软件设计中类之间的关系, 并使用一组形式化评估软件质量性质的定理进行评估, 结果表明, 新方法能够满足这些定理. 最后使用 JUnit 和 JEdit 作为研究对象, 利用 DependencyFinder 和 Eclipse 软件度量插件 Metrics 实现对软件耦合度量方法的自动计算, 计算结果验证了该方法的有效性.

**关键词:** C&K 度量组; 软件质量; 软件耦合度量; 统一建模语言类图

**中图分类号:** TP311

**文献标志码:** A

## Object-Oriented Software Coupling Metrics

MA Jian<sup>1</sup>, LIU Feng<sup>1</sup>, FAN Jian-ping<sup>2</sup>

(1. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China;

2. Shenzhen Institutes of Advanced Technology, Shenzhen 518055, China)

**Abstract:** The article analyzes the shortage of C&K software metrics suit for an improvement, then proposes a decomposed coupling metrics for objected-oriented (DCMOO). The metrics refer to unified modeling language class diagram to analyze the relationships between classes in software design, and evaluate the quality of the software by using a set of formal evaluation theorems, It is shown that DCMOO can satisfy these theorems. Finally the article uses JUnit and JEdit as research object, and applies DependencyFinder and Eclipse Metrics to calculate the proposed software coupling metrics automatically meanwhile validate the proposed metrics.

**Key words:** C&K metrics suit; software quality; software coupling metrics; unified modeling language class diagram

软件度量是一个范围很广的研究领域, 其中软件的设计度量是面向结构度量和面向对象度量的一个结合点. Chidamber 和 Kemerer 于 1994 年提出了 6 个面向对象设计的度量, 通常被称为 C&K 度量组. 软件设计度量研究中与软件设计水平紧密相关的耦合性度量研究是软件设计度量领域的一个研究热点. C&K 度量组本身是不完备的<sup>[1]</sup>, 度量组中耦合度量方面存在着许多不足, 例如: Hitz 和 Montazeri

等<sup>[2]</sup>认为 C&K 度量组类间的耦合 (CBO, coupling between object classes) 度量没有对耦合的强度做区分, 直接访问外部实例变量, 这通常被认定为是最差的耦合类型, 没有考虑继承耦合, 即访问从超类继承的实例变量. 这种耦合关系也被认为是最差的耦合类型之一, 此外, 没有考虑耦合的类型、耦合的扇入和扇出、接口的耦合. 统一建模语言 (UML, unified modeling language) 类图主要是用来显示系统中的

收稿日期: 2017-07-16

基金项目: 国家高技术研究发展计划 (863 计划) 项目 (2015AA043701)

作者简介: 马 健 (1985—), 女, 博士生, E-mail: 13112083@bjtu.edu.cn; 刘 峰 (1961—), 男, 教授, 博士生导师.

类、接口和它们之间的关系<sup>[3]</sup>,在UML类图中,类与类之间的关系可以反映设计时它们之间的耦合性.因此,参考UML类图对耦合度量进行分析是合理的,可以很好地解决上述问题. Julianade等采用多元logistic回归分析方法探讨了C&K度量组CBO耦合度量的综合效应,没有讨论UML类图分析类之间的耦合<sup>[4]</sup>. Badri等<sup>[5]</sup>提出了一种度量标准,即在不同场景下使用不同的度量指标.例如不同场景下C&K度量组CBO耦合度量,没有涉及UML类图分析类之间的耦合. Mathru等<sup>[6]</sup>使用UML类图分析类之间的耦合关系,但没有对耦合关系进行分解.

笔者针对上述问题,参考UML类图定义的软件系统类之间的关系,经过研究与分析,得到一种分解的软件耦合度量方法(DCMOO, decomposed coupling metrics for objected-oriented),可以反映面相对象设计中的耦合关系.

## 1 面向对象的软件耦合度量

### 1.1 C&K度量组描述

C&K度量方法给出了6个基于类的设计度量:

1) 每个类的加权方法数(WMC, weighted methods per class): 对于一个类 $C_i$ ,它定义了 $n$ 个方法 $M_1, M_2, \dots, M_n$ ,而 $c_1, c_2, \dots, c_n$ 是每个方法相应的复杂度,类的加权方法数为 $\sum_{i=1}^n c_i$

2) 继承树的深度(DIT, depth of inheritance tree): DIT是类在继承树中的深度,如果存在多继承的话,DIT是该类在继承数中的最大深度.

3) 子类的数目(NOC, number of children): NOC是类的直接子类的数目.

4) 类间的耦合(CBO, coupling between object classes): CBO是某个类存在耦合关系类的数量.

**定义1** 与该类存在耦合关系其他类的数量.

$$C = |\{d | u(c, d) \vee u(d, c)\}| \quad (1)$$

**定义2** 与该类存在耦合关系其他类的数量,特别是指出它不包含继承耦合关系.

$$C' = C - |\{d | a(c, d) \vee a(d, c)\}| \quad (2)$$

其中: $u(x, y)$ 定义了类 $x$ 的实现过程,使用了类 $y$ 的方法、实例或属性, $a(x, y)$ 定义了类 $x$ 是类 $y$ 的父类.

5) 类的响应(RFC, response for a class): 类的响应值为 $|RS|$ ,其中 $RS$ 是类对消息的响应集.响应集的值 $\{M\} \cup_{all} \{R_i\}$ , $\{M\}$ 是类的方法集, $\{R_i\}$ 是

被方法 $i$ 调用的方法集.

6) 类内聚缺乏度(LCOM, lack of cohesion methods)

$$L = \begin{cases} |P| - |Q|, & \text{若 } |P| > |Q| \\ 0, & \text{其他} \end{cases}$$

$P = \{(I_i, I_j) | I_i \cap I_j = \emptyset\}$ ,  $Q = \{(I_i, I_j) | I_i \cap I_j \neq \emptyset\}$ .  $\{I_i\}$ 是方法 $M_i$ 所用的实例变量集.

### 1.2 评估定理

文献[7]中描述了一组形式化评估软件质量性质的定理.

**性质1** 非粗糙性

$$(\exists P)(\exists Q)(\mu(P) \neq \mu(Q))$$

给定一个类 $P$ 和度量 $\mu$ ,总能找到另一个类 $Q$ 使得 $\mu(P) \neq \mu(Q)$ .表明并不是每个类的度量值都相同,否则该度量就失去了它的价值.

**性质2** 非唯一性

$$(\exists P)(\exists Q)(P \neq Q \rightarrow \mu(P) = \mu(Q))$$

可以存在不同的类 $P$ 和 $Q$ ,使得 $\mu(P) = \mu(Q)$ .这意味着2个不同的类可以有相同的度量值,即这2个类具有相同的复杂性.

**性质3** 设计细节重要性

$$(\exists P)(\exists Q)(P = Q \rightarrow \mu(P) \neq \mu(Q))$$

设2个类 $P$ 和 $Q$ 提供相同的功能,不意味着 $\mu(P) = \mu(Q)$ ,类必须影响度量值.

性质3表明,虽然两类执行相同的功能,但设计的细节决定了类的度量.

**性质4** 单调性:

$(\exists P)(\exists Q)(\mu(P) \leq \mu(P+Q) \wedge \mu(Q) \leq \mu(P+Q))$ 对所有类 $P$ 和 $Q$ ,必须有 $\mu(P) \leq \mu(P+Q)$ 和 $\mu(Q) \leq \mu(P+Q)$ ,其中 $P+Q$ 表示 $P$ 和 $Q$ 的组合.这意味着2个类度量值永远不会比其中任何一个小.

**性质5** 交互非等价性:

$(\exists P)(\exists Q)(\mu(P) = \mu(Q)) \rightarrow (\exists R)(\mu(P+R) \neq \mu(Q+R))$   $\exists P, \exists Q, \exists R$ ,使得 $\mu(P) = \mu(Q)$ 不表明 $\mu(P+R) = \mu(Q+R)$ , $P$ 和 $R$ 之间的相互作用不同于 $Q$ 和 $R$ 之间的相互作用导致 $P+R$ 和 $Q+R$ 不同的复杂度值.

## 2 DCMOO方法描述

C&K度量组中与耦合相关的度量有:NOC、CBO和RFC.这里重点讨论CBO,与文中提出的DCMOO度量进行比较.

C&K 度量组中的 DIT 和 NOC 度量可用于继承耦合, DCMOO 度量也包含继承耦合, 与 DIT 和 NOC 有功能上的重叠.

UML 类图中认为类图之间的主要关系有: 关联 (Association)、依赖 (Dependency)、泛化 (Generalization)、实现 (Realization). 其中, 耦合的强度为泛化 = 实现 > 关联 > 依赖.

参考 UML 中类图之间的关系, 结合面向对象软件的特点, 将类图中的泛化和实现关系合并, 从这 3 个方面分析耦合度量, 具体可分为: 关联耦合度量 (ACM, associated coupling metric)、依赖耦合度量 (DCM, dependent coupling metric) 和泛化耦合度量 (GCM, generalized coupling metric). 首先引入扇入和扇出耦合.

## 2.1 扇入扇出耦合度量

扇入扇出耦合: CBO 没有区别耦合关系的扇入和扇出, 首先对这种扇入和扇出耦合进行分析.

**定义 3** 类间的扇入耦合 (CBOin, coupling fan-in between object classes): 该类 (接口) 耦合其他类 (接口) 的数量.

$$I = |\{d | u(d, c)\}| \quad (3)$$

**定义 4** 类间的扇出耦合 (CBOout, coupling fan-out between object classes): 其他类 (接口) 耦合该类 (接口) 的数量.

$$O = |\{d | u(c, d)\}| \quad (4)$$

$u(x, y)$  定义了类 (接口)  $x$  的实现过程中, 使用了类 (接口)  $y$  的方法、实例或属性, 它包含继承耦合关系.

存在类 (接口)  $P, Q, R$ , 使得  $\mu(P) \neq \mu(Q)$  和  $\mu(P) = \mu(R)$ , 因此, 满足性质 1 和性质 2.

一个类 (接口) 与另一个类 (接口) 有耦合关系, 取决于耦合的方式并不是所提供的功能, 因此性质 3 是满足的.

设  $P$  和  $Q$  是 2 个类 (接口), 并且  $\mu(P) = n_p$ ,  $\mu(Q) = n_q$ , 如果  $P$  和  $Q$  合并, 就会有  $\mu(P + Q) = n_p + n_q - \delta$ , 其中,  $\delta$  是  $P$  和  $Q$  合并后减少的耦合关系数. 显然,  $n_p - \delta \geq 0$ ,  $n_q - \delta \geq 0$ , 耦合减少的不会比原来耦合的多, 因此, 对于所有的  $P$  和  $Q$ ,  $n_p + n_q - \delta \geq n_p$ ,  $n_p + n_q - \delta \geq n_q$ , 性质 4 满足.

设  $P$  和  $Q$  是 2 个类 (接口),  $\mu(P) = \mu(Q) = n$ , 设  $R$  是另一个类 (接口),  $\mu(R) = r$ , 有  $\mu(P + Q) = n + r - \delta$ ,  $\mu(Q + R) = n + r - \beta$ ,  $\delta$  和  $\beta$  是相互独立的, 不会永远相等, 所以  $\mu(P + R) \neq \mu(Q + R)$ . 满足性

质 5.

在扇入扇出耦合度量的基础上, 进一步细化为关联耦合度量 (ACM)、依赖耦合度量 (DCM)、泛化耦合度量 (GCM).

## 2.2 关联耦合度量

**定义 5** 关联耦合度量: ACM 是软件系统中类 (接口)  $c$  所关联的类 (接口) 的数量.

$$A = |\{d | T(a) = d \wedge a \in F(c)\}| \quad (5)$$

其中:  $F(c)$  表示类 (接口)  $c$  的成员变量的集合,  $T(a)$  表示成员变量  $a$  的类型.

存在类 (接口)  $P, Q, R$ , 使得  $\mu(P) \neq \mu(Q)$  和  $\mu(P) = \mu(R)$ , 因此, 满足性质 1 和性质 2.

类 (接口) 之间有关联关系时, 一个类 (接口) 的成员变量的类型为另一个类 (接口), 取决于关联的方式而不是所提供的功能, 因此性质 3 是满足的.

设  $P$  和  $Q$  是两个类 (接口) 并且  $\mu(P) = n_p$ ,  $\mu(Q) = n_q$ , 如果类 (接口)  $P$  和  $Q$  合并, 就会有  $\mu(P + Q) = n_p + n_q - \delta$ , 其中,  $\delta$  是  $P$  和  $Q$  合并后减少的关联耦合数, 显然,  $n_p - \delta \geq 0$ ,  $n_q - \delta \geq 0$ , 因此, 关联中减少的不比原来关联的数量多. 对于所有的  $P$  和  $Q$ ,  $n_p + n_q - \delta \geq n_p$ ,  $n_p + n_q - \delta \geq n_q$ , 性质 4 满足.

设  $P$  和  $Q$  是 2 个类 (接口),  $\mu(P) = \mu(Q) = n$ , 设  $R$  是另一个类 (接口),  $\mu(R) = r$ , 有  $\mu(P + Q) = n + r - \delta$ ,  $\mu(Q + R) = n + r - \beta$ ,  $\delta$  和  $\beta$  是相互独立的, 不会永远相等, 所以  $\mu(P + R) \neq \mu(Q + R)$ . 满足性质 5.

## 2.3 依赖耦合度量

**定义 6** 依赖耦合度量: DCM 是软件系统中类 (接口)  $c$  所依赖的类 (接口) 的数量.

$$D = |\{d | a \in P(m) \wedge T(a) = d\}| \quad (6)$$

其中:  $P(m)$  表示方法  $m$  的参数集、返回值和局部变量,  $T(a)$  表示成员变量  $a$  的类型.

存在类 (接口)  $P, Q, R$ , 使得  $\mu(P) \neq \mu(Q)$  和  $\mu(P) = \mu(R)$ , 因此, 满足性质 1 和性质 2.

类 (接口) 之间有依赖关系时, 一个类 (接口) 的函数参数局部变量、参数或返回值的类型为另一个类 (接口), 取决于依赖的方式并不是所提供的功能, 因此性质 3 是满足的.

设  $P$  和  $Q$  是 2 个类 (接口), 并且  $\mu(P) = n_p$ ,  $\mu(Q) = n_q$ , 如果类 (接口)  $P$  和  $Q$  合并, 就会有  $\mu(P + Q) = n_p + n_q - \delta$ . 其中,  $\delta$  是  $P$  和  $Q$  合并后减少的依赖耦合数, 显然,  $n_p - \delta \geq 0$ ,  $n_q - \delta \geq 0$ , 因此,

依赖中减少的不比原来依赖的数量多,对于所有的  $P$  和  $Q$ ,  $n_p + n_Q - \partial \geq n_p, n_p + n_Q - \partial \geq n_Q$ , 性质 4 满足.

设  $P$  和  $Q$  是 2 个类(接口),  $\mu(P) = \mu(Q) = n$ . 设  $R$  是另一个类(接口),  $\mu(R) = r$ , 有  $\mu(P + Q) = n + r - \partial, \mu(Q + R) = n + r - \beta, \partial$  和  $\beta$  是相互独立的, 不会永远相等, 所以  $\mu(P + R) \neq \mu(Q + R)$ . 满足性质 5.

2.4 泛化耦合度量

**定义 7** 泛化耦合度量: GCM 是软件系统中类  $c$  所继承(实现)类(接口)的集合或接口  $c$  所继承接口的数量.

$$G = |\{d | d \in R(c)\}|$$

(7)

其中  $R(c)$  表示类(接口)  $c$  的父类(接口)集合或类  $c$  实现的接口.

存在类(接口)  $P, Q, R$ , 使得  $\mu(P) \neq \mu(Q)$  和  $\mu(P) = \mu(R)$ , 因此, 满足性质 1 和性质 2.

类(接口)之间有泛化关系时, 一个类(接口)的方法使用另一个类(接口)类型的成员变量和函数成员, 取决于泛化的方式并不是所提供的功能, 因此性质 3 是满足的.

设  $P$  和  $Q$  是 2 个类(接口), 并且  $\mu(P) = n_p, \mu(Q) = n_Q$ , 如果  $P$  和  $Q$  合并, 就会有  $\mu(P + Q) = n_p + n_Q - \partial$ . 其中,  $\partial$  是  $P$  和  $Q$  合并后减少的泛化耦合数. 显然,  $n_p - \partial \geq 0, n_Q - \partial \geq 0$ . 因此, 泛化中减少的不比原来泛化的数量多, 对于所有的  $P$  和  $Q$ ,  $n_p + n_Q - \partial \geq n_p, n_p + n_Q - \partial \geq n_Q$ , 性质 4 满足.

设  $P$  和  $Q$  是 2 个类(接口),  $\mu(P) = \mu(Q) = n$ , 设  $R$  是另一个类(接口),  $\mu(R) = r$ , 有  $\mu(P + Q) = n + r - \partial, \mu(Q + R) = n + r - \beta, \partial$  和  $\beta$  是相互独立的, 不会永远相等, 所以  $\mu(P + R) \neq \mu(Q + R)$ . 满足性质 5.

3 实验结果与分析

JUnit 是一个 Java 语言的单元测试框架, JEdit 是一个用 Java 语言开发的文本编辑器. 笔者使用 JUnit4.3 和 JEdit5.4 两个版本作为研究对象分析了度量值的计算结果.

实验使用了软件度量工具 DependencyFinder 和度量软件插件 Eclipse Metrics 实现对软件耦合度量值的计算.

表 1 和表 2 所示为计算得出的 JUnit4.3 和 JEdit5.4 的 CBO 度量值.

表 1 JUnit4.3 的 CBO 度量值

CBO	JUnit4.3	占比/%
1 ~ 10	192	85.00
11 ~ 20	20	8.85
21 ~ 30	8	3.54
31 ~ 40	5	2.21
41 ~ 50	1	0.44
> 50	0	0

表 2 JEdit5.4 的 CBO 度量值

CBO	JEdit5.4	占比/%
1 ~ 10	991	81.8
11 ~ 20	141	11.6
21 ~ 30	38	3.14
31 ~ 40	14	1.16
41 ~ 50	10	0.83
> 50	18	1.46

大部分类的耦合度量值较小, 只有少数类的耦合度量值较大. 对象之间的过度耦合不利于模块的设计和重用. 一个类越是相对独立, 就越容易被重用用于其他系统中.

表 3 和表 4 所示为 JUnit4.3 和 JEdit5.4 的扇入/扇出耦合度量值.

表 3 JUnit4.3 的扇入/扇出耦合度量值

CBOout/ CBOin	CBOout		CBOin	
	JUnit4.3	占比/%	JUnit4.3	占比/%
0 ~ 10	219	91.6	226	94.6
11 ~ 20	13	5.44	11	4.60
21 ~ 30	5	2.09	2	0.84
31 ~ 40	1	0.41	0	0
41 ~ 50	0	0	0	0
> 50	1	0.41	0	0

表 4 JEdit5.4 扇入/扇出耦合度量值

CBOout/ CBOin	CBOout		CBOin	
	JEdit5.4	占比/%	JEdit5.4	占比/%
0 ~ 10	1 126	91.1	1 135	91.9
11 ~ 20	83	6.72	57	4.61
21 ~ 30	20	1.62	13	1.05
31 ~ 40	4	0.32	13	1.05
41 ~ 50	2	0.16	5	0.04
> 50	1	0.08	13	1.05

通过分析 JUnit 和 JEdit 的度量值方法可以得出

以下结论：

- 1) 大部分类(接口)的扇入耦合和扇出耦合的度量值较小,只有少数类(接口)的扇入扇出耦合值较大；
- 2) 扇出耦合较小,但扇入耦合较大的类一般都为接口；
- 3) 为了提高模块化和封装性,应该尽量减少类(接口)之间的耦合. 耦合关系的数量越大,类(接口)也越容易受到其他部分改动的影响,因而维护也越困难.

表 5 和表 6 所示为 JUnit4. 3 和 JUnit5. 4 的关联耦合度量值.

表 5   JUnit4. 3 的关联耦合度量值

ACM	JUnit4. 3	占比/%
0	216	89. 9
1	16	7. 02
2	5	2. 19
3	1	0. 44
4	0	0
>4	1	0. 44

表 6   JUnit5. 4 的关联耦合度量值

ACM	JUnit5. 4	占比/%
0	965	78. 7
1	151	12. 2
2	56	4. 53
3	36	2. 91
4	16	1. 29
>4	12	0. 97

通过分析 JUnit 和 JUnit5 的度量值可以得出以下结论：

- 1) 大部分类(接口)的关联度量值都较小；
- 2) 关联关系不利于模块化设计,减少关联关系有利于程序的修改、升级和测试；
- 3) 为了提高模块化和封装性,降低耦合强度,使得关联关系变得很少.

表 7 和表 8 所示为 JUnit4. 3 和 JUnit5. 4 的依赖耦合度量值.

通过分析 JUnit 和 JUnit5 的度量值方法可以得出以下结论：

- 1) 大部分类(接口)的依赖度量值都较大；
- 2) 依赖关系有利于模块化设计,有利于程序的

扩展和重用,是一种较弱的耦合关系；

表 7   JUnit4. 3 的依赖耦合度量值

DCM	JUnit4. 3	占比/%
0 ~ 10	223	93. 0
11 ~ 20	15	6. 58
22 ~ 30	1	4. 39
31 ~ 40	0	0
41 ~ 50	0	0
>50	0	0

表 8   JUnit5. 4 的依赖耦合度量值

DCM	JUnit5. 4	占比/%
0 ~ 10	1 155	93. 4
11 ~ 20	65	5. 25
21 ~ 30	12	0. 97
31 ~ 40	3	0. 24
41 ~ 50	1	0. 09
>50	0	0

- 3) 为了提高封装性,降低耦合强度,使依赖关系增加,系统模块易于重用,系统容易扩展.

表 9 和表 10 所示为 JUnit4. 3 和 JUnit5. 4 的泛化耦合度量值.

表 9   JUnit4. 3 的泛化耦合度量值

GCM	JUnit4. 3	占比/%
0	203	84. 2
1	21	9. 21
2	8	3. 51
3	5	2. 20
4	0	0
>4	2	0. 88

表 10   JUnit5. 4 的泛化耦合度量值

GCM	JUnit5. 4	占比/%
0	802	64. 9
1	407	92. 9
2	25	2. 02
3	1	0. 08
4	0	0
>4	1	0. 08

通过分析 JUnit 和 JUnit5 的度量值方法可以得出以下结论：

- 1) 大部分类(接口)的泛化度量值都较小；
- 2) 泛化关系本身就是较强的耦合关系,系统中的继承为子类(接口)继承父类(接口),一个类实现



另一个接口;

3) 为了提高多态性和封装性,降低耦合强度,使得泛化关系变得较少.

## 4 结束语

分析了 C&K 度量组中类间耦合 CBO 的不足,参考 UML 中类图之间的关系,对 CBO 度量指标进行了改进,并使用一组形式化评估软件质量性质的定理进行评估;以 JUnit 和 JEdit 为研究对象,对提出度量框架的关联、依赖、泛化关系进行度量研究. 结果分析表明,DCMOO 可以较准确地反映面向对象设计中的耦合关系. 设计良好的面向对象系统,其面向对象设计度量的值也较为合理;反之,设计较差的面向对象系统,其面向对象设计度量的值也并不甚合理,从而验证了方法的有效性.

新方法的提出揭示了面向对象开发设计的一定规律,对软件产品的解耦合和维护升级也有一定的指导意义.

## 参考文献:

- [1] Chidamber S R, Kemerer C F. A metrics suite for object oriented design[J]. IEEE Transactions on Software Engineering, 1994, 20(6): 476-493.
- [2] Hitz M, Montazeri B. Measuring coupling and cohesion

in object-oriented system[C] // Proceedings of International Symposium on Applied Corporate Computing (ISAAC 95). Mexico: Springer, 1995: 78-84.

- [3] 冯跃忠, 李晓峰. 基于规则库的电信业务 UML 活动图验证机制[J]. 北京邮电大学学报, 2008, 31(2): 76-79.  
Feng Yuezhong, Li Xiaofeng. A rule database based UML activity model checking mechanism in telecommunication service[J]. Journal of Beijing University of Posts and Telecommunications, 2008, 31(2): 76-79.
- [4] Juliana de A G Saraiva, MicaelS de França, Sérgio C B Soares, et al. Classifying metrics for assessing object-oriented software maintainability: a family of metrics' catalogs[J]. Journal of Systems & Software, 2015, 103(C): 85-101.
- [5] Badri M, Toure F. Empirical analysis of object-oriented design metrics for predicting unit testing effort of classes[J]. Journal of Software Engineering & Applications, 2014, 5(7): 513-526.
- [6] Mathru B, Kaushik M. Empirical analysis of metrics using UML class diagram[J]. International Journal of Advanced Computer and Applications, 2016, 7(5): 32-37.
- [7] Srinivasan K P, Devi T. Software metrics validation methodologies in software engineering[J]. International Journal of Software Engineering & Applications(IJSEA), 2014, 5(6): 87-102.