ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.17, No.4, April 2006, pp.703−712
DOI: 10.1360/jos170703

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 一种自动化软件设计改进方法*

冯 铁+, 张家晨, 王洪媛, 金淳兆

(吉林大学 计算机科学与技术学院 符号计算与知识工程教育部重点实验室,吉林 长春 130012)

## An Approach to Automated Software Design Improvement

FENG Tie+, ZHANG Jia-Chen, WANG Hong-Yuan, JIN Chun-Zhao

(Key Laboratory Symbolic Computation and Knowledge Engineering of Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun 130012, China)

+ Corresponding author: Phn: +86-431-8829771, E-mail: fengtie@jlu.edu.cn

**Feng T, Zhang JC, Wang HY, Jin CZ. An approach to automated software design improvement. *Journal of Software*, 2006,17(4):703−712.** http://www.jos.org.cn/1000-9825/17/703.htm

**Abstract**: Object-Oriented software design improving technology is an effective means to increase system flexibility for adapting to future requirement variation and expansion. In this paper, a software design improving approach, based on micro-architecture anti-pattern and case based reasoning, is presented to improve software quality and maintainability. In this approach, problematic, inflexible structures and corresponding refactoring alternatives at micro-architecture level are formally defined and described as cases. Their organization and index mechanism in the case base are studied. Following the 4R procedures of CBR, similarity measurement methods on class diagrams, sequence diagrams, OO quality metric factors, and semantic constraints are discussed. Based on the measurement results, some algorithms on identifying anti-patterns instances in a given original design and replacing them by designs with high quality are presented. Furthermore, a supporting system CBDIT is developed to aid this approach.

**Key words**: case based reasoning; design pattern; anti-pattern; software evolution; refactoring

摘 要: 面向对象的软件设计改进是增强系统的可扩展性、使之适应可能的需求变化的一种有效手段.提出一种基于 CBR(case based reasoning)和微体系结构反模式的设计改进方法.该方法形式化地定义了在微体系结构层中不灵活的设计结构和相应的重构方案的描述方法,研究了它们在事例库中的组织和索引机制.根据基于事例的推理技术的 4R 过程,提出了类图、顺序图、质量要素和语义约束的相似性度量方法,描述了在给定设计中识别反模式及其实例,并在此基础上,用高质量的设计方案进行替代的几种算法.以该方法为依据,进一步介绍了设计改进支撑系统——CBDIT(CBR based design improving tool)的体系结构的设计.

关键词: 基于事例的推理;设计模式;反模式;软件演化;重构

中图法分类号: TP311 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.69903005 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA115160 (国家高技术研究发展计划(863)); the "985 Project" of Jilin University of China (吉林大学 985 项目); the Innovation Foundation of Jilin University of China (吉林大学创新基金)

Received 2005-03-29; Accepted 2005-10-10

在当今软件开发领域中一个亟待解决的问题是:处理软件产品在交付使用后的变化和扩展.产生这种变动的原因,除已开发的产品与原始需求之间固有的偏差之外,也来自于系统运行环境的改变和用户对系统功能及性能需求的增加.处理这种变动的困难在于:许多遗产系统在微体系结构层的设计不够灵活.因此,在给定的设计中识别这些刻板的设计结构,并将它们用资深专家建议的方案或经过实践验证的高质量设计进行替代,对于提高软件可维护性和软件质量具有重要的意义.

作为表达面向对象设计经验的一种有效机制,设计模式对面向对象设计原理进行识别、命名和抽象.它通过标识对象、对象间的合作及责任分配来揭示设计机理[1].反模式是对设计模式研究的延伸,它以文字形式描述在解决某问题时反复出现的一种产生负面效果的方案[2].经过某种精化或实例化,模式和反模式具有良好的粒度和抽象度,很容易被应用于软件设计或从原始设计中识别出来.但是,它们通常采用一种基于自然语言的描述方式.因此,有必要研究和开发某种方法工具,自动化识别低质量设计,并用高质量设计进行替换.

另一方面,人工智能技术正被广泛应用于工程实践活动中.作为一种有效的问题求解和机器学习技术,CBR (case based reasoning)同样适用于软件工程的特定活动——设计改进之中,原因如下:① CBR 能够记忆和利用先前所经历过的、具体的问题空间的知识;② CBR 是一种渐增的、稳定的学习技术;③ CBR 是基于大块知识的推理方法;④ CBR 易于作为推理框架,集成其他人工智能技术.

本文第 1 节详细描述微体系结构反模式,并用一个例子解释这个概念.第 2 节讨论应用 CBR 进行设计改进的方法,包括事例的表示、检索、事例库组织和索引、事例相似性估算、事例的调整和重用,我们也提供了几种算法.第 3 节介绍基于上述思想开发的支撑系统 CBDIT(CBR based design improving tool),讨论与该环境的开发相关的几个问题.第 4 节讨论与本文研究相关的工作.最后,我们得出结论并指出未来的工作.

## 1 微体系结构反模式

软件设计包括 7 个体系结构层次:全球级、企业级、系统级、应用级、宏组件级(框架级)、微体系结构级和对象级.微体系结构级设计的一个重要特征是:由一组协同工作的对象构成,对象间的相互关联由该组件的实现者定义和理解.微体系结构能够处理一个或多个问题域中的一系列问题[2].

微体系结构反模式是在该抽象层次上的具有负面效果的设计,这也是我们希望从给定的原始设计中识别的内容.但是,这个非形式化的定义对于自动化匹配是不够的,所以我们首先给出反模式的定义.

如果一个体系结构 $D$ 满足下列 3 个条件,则称 $D$ 是一个微体系结构反模式*:

(1) $D=\langle StaticStructure\rangle|\langle StaticStructure\rangle\langle DynamicBehavior\rangle$

$StaticStructure=\langle SClasses\rangle\langle SRelationships\rangle$

$SClasses=\langle Class\rangle^*$

$Class=\langle ClassName\rangle|\langle ClassName\rangle\langle Attribute\rangle^*\langle Method\rangle^*$

$SRelationship=\langle Relation\rangle^*$

$Relation=\langle Inh\rangle|\langle Agg\rangle|\langle Ini\rangle|\langle Del\rangle$

$Inh=$“$Inh$(”$\langle ClassName\rangle$“,”$\langle ClassName\rangle$“)”

$Agg=$“$Agg$(”$\langle ClassName\rangle$“,”$\langle ClassName\rangle$“,”$\langle Multiple\rangle$“)”

$Ini=$“$Ini$(”$\langle ClassName\rangle$“,”$\langle ClassName\rangle$“)”

$Del=$“$Del$(”$\langle ClassName\rangle$“,”$\langle ClassName\rangle$“)”

$DynamicBehavior=\langle IS\rangle^*$

$IS=$“(”$\langle io\rangle$“,”$\langle ro\rangle$“,”$\langle msg\rangle$”)”

$io=\langle ObjectName\rangle$

$ro=\langle ObjectName\rangle$

* 由于篇幅有限,这里省略了一些含义明确的非终极符的定义细节.

*msg*=⟨*ReturnType*⟩⟨*MsgName*⟩"("⟨*FormalParameterList*⟩")"

(2) 不少于 3 次的系统开发实践证明 *D* 会导致性能和可扩展性方面的问题;

(3) *D* 中不超过 8 个类或对象.**

我们认为,反模式主要由静态结构设计和动态行为设计两个方面构成:静态结构设计结果中主要包含 4 种关系:*Inh*,*Agg*,*Ini* 和 *Del*,即继承、聚合、实例化和委托.我们认为,关联、组合、依赖等关系主要应用于在分析阶段理解问题域,它们在设计阶段都可以转换成上述的 4 种关系;系统动态行为的设计通常用顺序图或合作图表示.为了识别不良系统动态行为,如与数据库进行的过度频繁的交互、对象间为保持内容一致性而频繁发送消息等等,我们把对象间的交互定义为交互序列.一个交互序列是一个元组列表,每个元组由 3 个元素组成:发起对象、响应对象和消息签名.

· 一个反模式实例的例子

图 1 表示了一个反模式实例,这个设计描述了系统的静态结构,从而满足了反模式定义的第一项要求.类 StaffManagement 通过聚合关系与类 Student,Teacher 以及 Dushman 联系在一起,方法 NewTerm 通过判断当前所处理的对象标识,对这 3 个类的、语义上十分类似的行为 Register,Enroll 和 CheckIn 进行调用.假设用户要求增加对较长的一种特定的注册处理,如增加类 Principal 及操作 Login,那么维护人员必须分析类 StaffManagement 的方法 Newterm,在数十甚至数百行代码中找到适当的位置添加代码,完成对 Principal 的方法 Login 的调用.因此,该设计典型地代表了一个反复出现的不灵活的、不利于扩展的负面设计结果,从而满足了反模式定义的第 2 项要求.该设计是一个反模式的具体实例,类的个数没有限制.

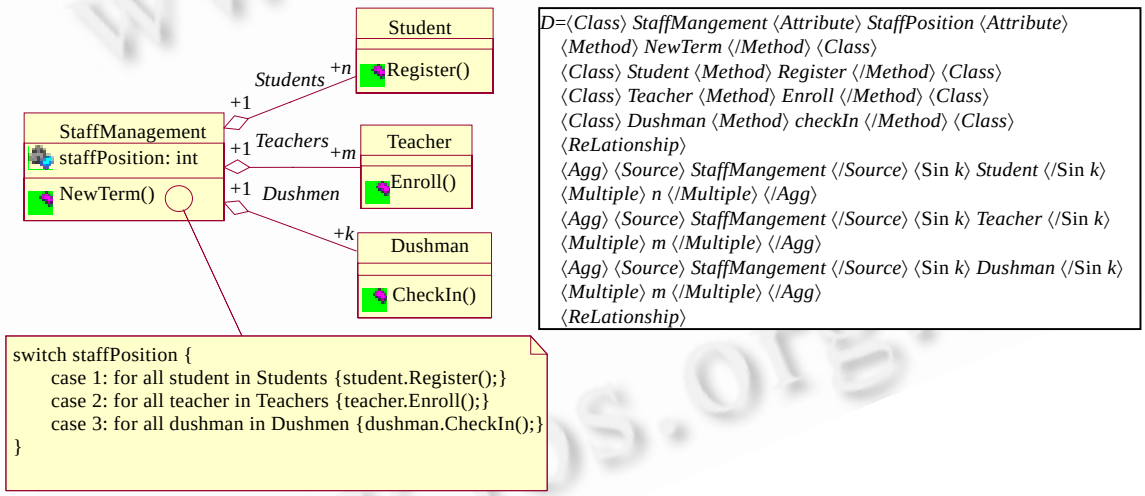


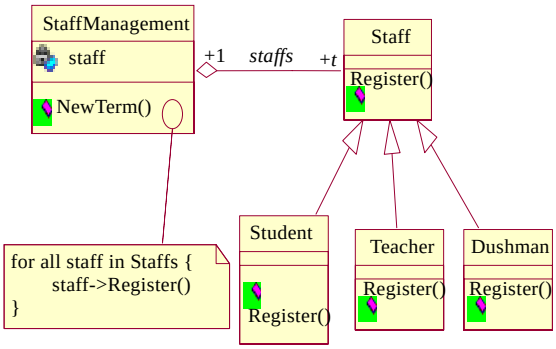

Fig.1 An example of anti-pattern instance and the corresponding XML representation

图 1 一个反模式实例的例子及其相应的 XML 表示

· 设计改进后的期望结果

图 2 表示了上述设计在应用本文提出的方法和工具进行自动化设计改进后的结果.该设计引入了一个新类 Staff 及其方法 Register,类 StaffManagement 引入 staff 属性并通过它与类 Staff 联系,NewTerm 方法仅访问 staff 属性.Student,Teacher 和 Dustman 通过重写 Register 方法完成不同的注册过程.这种设计充分利用了面向对象的多态和动态联编特性.这种设计的优点在于,对于新增加的需求,改进后的设计只需从 Staff 类继承一个新类 Princinpal 并重写 Register 方法,最后将 staff 属性赋值为新增类的对象即可.

** 微体系结构反模式实例与微体系结构反模式不同,它可以超过 8 个类.




```
D=⟨Class⟩ StaffMangement ⟨Attribute⟩ StaffPosition ⟨Attribute⟩
  ⟨Method⟩ NewTerm ⟨/Method⟩ ⟨Class⟩
  ⟨Class⟩ Student ⟨Method⟩ Register ⟨/Method⟩ ⟨Class⟩
  ⟨Class⟩ Teacher ⟨Method⟩ Enroll ⟨/Method⟩ ⟨Class⟩
  ⟨Class⟩ Dushman ⟨Method⟩ checkIn ⟨/Method⟩ ⟨Class⟩
  ⟨ReLationship⟩
  ⟨Agg⟩ ⟨Source⟩ StaffMangement ⟨/Source⟩ ⟨Sin k⟩ Student ⟨/Sin k⟩
  ⟨Multiple⟩ n ⟨/Multiple⟩ ⟨/Agg⟩
  ⟨Agg⟩ ⟨Source⟩ StaffMangement ⟨/Source⟩ ⟨Sin k⟩ Teacher ⟨/Sin k⟩
  ⟨Multiple⟩ m ⟨/Multiple⟩ ⟨/Agg⟩
  ⟨Agg⟩ ⟨Source⟩ StaffMangement ⟨/Source⟩ ⟨Sin k⟩ Dushman ⟨/Sin k⟩
  ⟨Multiple⟩ m ⟨/Multiple⟩ ⟨/Agg⟩
  ⟨ReLationship⟩
```

Fig.2 Improved design and the corresponding XML representation

图 2 改进后的设计及其相应的 XML 表示

## 2 在设计改进中应用基于事例的推理

基于事例的推理是基于过去经验的一种推理方法,通常包括 4 个步骤:取回(retrieve)、重用(reuse)、修正(revise)和保存(retain),这也称为 CBR 的 4R 过程.本节讨论事例的表示、事例库的组织和检索、事例的重新应用、修正性的学习和保存.事例表示方法可以分为两种:基于特征值的表示和基于图形的表示[3].在传统的 CBR 方法中,事例通常用属性元组表示.两个事例的相似度使用特征值距离度量,每个特征值对应不同的权值.形式化地, $Similarity(Case_m, Case_r) = N\left(1-\sum_{i=1}^{n} w_i \times Dis(Case_m.Attribute[i], Case_r.Attribute[i])\right)$.其中,$Dis$ 是距离函数,它根据$Attribute[i]$的类型定义; $w_i$ 是根据每个属性对检索的重要性不同而赋予 $Attribute[i]$的权值;$N$是正规化函数[4].在更为复杂的应用领域,事例的基本元素表示为节点,元素间的关系表示成弧.节点和弧上加以标记,这些标记通常是一元或二元的谓词.根据面向对象设计领域的特点,我们的方法集成了基于特征值的事例描述形式和基于图形的事例描述形式,用综合的方式表示软件设计改进事例.

### 2.1 事例表示

设计改进事例标识改进前后设计方案的结构、行为及其他细节信息,改进前后软件质量要素的值.一个软件设计改进事例包含 5 个主属性:反模式描述(APD)、语义约束(SC)、改进前质量度量(QMBI)、模式描述(PD)、改进后的质量度量(QMAI),每个主属性由一些子属性或图形结构组成.

#### 2.1.1 反模式描述和模式描述

APD 和 PD 都包括 4 个子属性:名字、应用结果、静态结构和动态行为.其中,静态结构和动态行为是相似性度量的两个重要的参考指标,这两个子属性有两种存储形式:XML 文件及 SQL 数据库,以及 4 种描述形式:UML 图、结构图形、XML 文档和邻接表;另外的两个子属性,名称和应用结果主要采用文本形式用于事例库的组织和检索空间的缩小.

定义 **2.1**. 一个类结构图定义为一个四元组,记为 $G=(N_c,E_r,\beta,L_\beta)$,其中,$N_c$ 是有限的节点集合,每个节点代表一个类;$E_r \subseteq N_c \times N_c$ 是有限的边集合,每条边代表一个关系;$\beta: E_r \to L_\beta$是边的标记映射函数,$L_\beta \subseteq \{Inh,Agg,Del,Ini\}$是边上的标记.

定义 **2.2**. 一个对象结构图定义为一个四元组,记为 $G=(N_0,E_m,\beta,L_\beta)$.其中,$N_0$ 是有限的节点的集合,每个节点代表一个对象;$E_r \subseteq N_0 \times N_0$ 是有限的边集合,每条边代表一个消息;$\beta: E_m \to L_\beta$是边的标记映射函数,$L_\beta \subseteq PosInt$, $PosInt=\{x | x \in Int \wedge x \succ 0\}$是边上的标记.

很明显,类结构图和对象结构图都是有向标记图的特例,唯一的不同在于我们省略了节点标记和节点标记映射函数.这样做可以保证通过对类和对象的重命名机制获得相似的事例.在这个阶段,只关心类和对象的结构信息.

定义 **2.3**. 令 $G_t=(N_t,E_t,\beta_t,L_\beta)$和 $G_s=(N_s,E_s,\beta_s,L_\beta)$是两个类结构图或对象结构图,我们称存在从 $G_t$ 到 $G_s$ 的一个映射 $\eta:G_t\to G_s$,如果有:

(1) 对于$\forall n_t\in N_t$,$n_t$ 与一个且仅与一个 $n_s\in N_s$ 对应,记作 $\eta(n_t)=n_s$;

(2) 对于$\forall e_t=(n_{t1},n_{t2})\in E_t$,如果 $\eta(n_{t1})=n_{s1}$,$\eta(n_{t2})=n_{s2}$ 并且 $e_s=(n_{s1},n_{s2})\in E_s$,则 $e_t$ 与 $e_s$ 对应,记作 $\eta(e_t)=e_s$.

从上述定义可以看出,如果存在一个从 $G_t$ 到 $G_s$ 的映射 $\eta:G_t\to G_s$,则 $G_t$ 中所包含的节点数目不大于 $G_s$ 中所包含的节点数目.利用这一特性,我们试图找到从反模式描述所对应的类结构图或对象结构图到原始设计所对应的类结构图或对象结构图的所有映射.

类图中的不同关系及顺序图中不同的消息在表达系统静态结构和动态行为方面具有不同的重要性.相应地,我们为反模式描述中的类结构图和对象结构图中的每条边赋予不同的权值,并在相似性度量时使用这些权值.它们也会在学习的过程中动态调整.

定义 **2.4**. 令 NEC 是反模式描述中类结构图中边的个数,每条边的缺省权值定义为 $0.3\times(1/NEC)$.

定义 **2.5**. 令 NEO 是反模式描述中对象结构图中边的个数,每条边的缺省权值定义为 $0.2\times(1/NEO)$.

2.1.2 改进前后的质量因素度量值

改进前后的质量因素度量值都包括 8 个数字类型子属性:每个类的加权方法(WMC)、子类数目(NOC)、继承树的深度(DIT)、类响应数(RFC)、对象间耦合(CBO)、方法间的内聚缺乏程度(LCOM)、抽象类的个数(NOAC)以及与抽象类有关的关系数(NRWAC),前 6 个质量因素在文献[5]中已有定义.我们认为:在微体系结构设计中,抽象类的个数以及与抽象类有关的关系数反映了多态机制的应用情况,也就反映了该设计扩展和变动的灵活程度.例如,表 1 列出了图 1 和图 2 中设计的 NOAC 值和 NRWAC 值.

**Table 1** Changes on NOAC and NRWAC before and after improving

表 **1** 改进前后 NOAC 和 NRWAC 的变化

| | Design in Fig.1 | Design in Fig.2 |
|---|---|---|
| NOAC | 0 | 1 |
| NRWAC | 0 | 4 |

2.1.3 语义约束

即使存在一个从 $G_t$ 到 $G_s$ 的映射 $\eta:G_t\to G_s$,我们也无法断定对原始设计引入变动的必要性.换言之,缺乏语义信息的结构匹配是设计改进的必要非充分条件.

静态结构的语义约束是描述设计知识的加权逻辑谓词.它们是具体上下文相关的,也即与具体的设计内容有关,所以不能像处理图形映射那样统一处理.例如,图 1 和图 2 所表示的设计改进事例的语义约束如下:

$$MeaningSimilar(Class_{i1}.Function_{i1},Class_{i2}.Function_{i2},\ldots,Class_{im}.Function_{im}) \quad [\omega 1]$$

$$Invoke(Class_z.Function_y,Class_{j1}.Function_{j1},\ldots,Class_{jm}.Function_{jm}) \quad [\omega 2]$$

$$Class_{i1}.Function_{i1}=Class_{j1}.Function_{j1}\wedge\ldots\wedge Class_{im}.Function_{im}=Class_{jm}.Function_{jm} \quad [\omega 3]$$

满足上述谓词的事实如下:

$$MeaningSimilar(Student.Register,Teacher.Enroll,DushMan.CheckIn) \quad [\omega 1]$$

$$Invoke(StaffManagenent.NewTerm,Student.Register,Teacher.Enroll,DushMan.CheckIn) \quad [\omega 2]$$

$$Student.Register=Student.Register\wedge Teacher.Enroll$$
$$=Teacher.Enroll\wedge DushMan.CheckIn=DushMan.CheckIn \quad [\omega 3]$$

定义 **2.6**. 令 *NSC* 为语义约束的数目,每个谓词的缺省权值定义为 $0.3\times(1/NSC)$.

例如,*MeaningSimilar*,*Invoke* 和等式的缺省权值都是 $0.3\times(1/3)=0.1$.在判定谓词时常常需要访问同义词库,如 MeaningSimilar 谓词的判定,同义词库根据领域名称和词性组织和索引.我们的方法考虑并支持 8 个领域的设计改进:制造业、电信业、医疗领域、保险业、财务领域、旅游业、电子商务、职业服务,因为 Len Silverston[6] 已经对这 8 个领域开发了较为完整的数据模型.

对象结构图由顺序图演化而来,故可以使用 UML 的对象约束语言 OCL[7],沿用 OCL 的语义来约束对象的

动态交互.通常情形下,只需为对象结构图填加对象交互时序约束,便可以完成比较精确的匹配结果.

### 2.2 事例库的组织和索引

事例库按照层次模型进行组织和索引.根据 APD 的名称查同义词库,为所查找事例赋以一个描述子 $X(X=G(general)|M(Manufacturing)|T(Telecommunication)|H(Health\ care)|I(Insurance)|F(Financial)|P(Professional\ service)|R(tRavel)|E(E\text{-}commerce))$,$X$ 指明一个特定领域或一般领域.根据类结构图的相似性匹配结果,搜索范围进一步缩减到一个较小的事例集,之后在原有的描述子后添加一个新的描述子 $C_i$,依此类推,可以找到一条到多条从根到某个或某些语义约束的路径,同时也形成了一个或多个完整的描述子 $XC_iOjQ_mS_n$.其中,$C$ 代表类结构图的描述子;$O$ 代表对象结构图的描述子;$Q$ 代表质量要素的描述子;$S$ 代表语义约束的描述子.这里,我们使用 XML 将事例层次化地组织和存储在事例库中.这样易于将来对系统功能的进一步扩展.

### 2.3 事例检索

根据事例的表示形式和事例库的组织结构,事例的检索包含 4 个步骤:类结构图匹配、对象结构图匹配、质量度量数据匹配和语义约束匹配.根据历史数据所显示的上述 4 个部分在事例检索中的重要性,在我们的方法中,这 4 部分的相似度($DS_c$,$DS_o$,$DS_q$,$DS_s$)的最大值分别规定为:0.3,0.2,0.3 和 0.2,即 $\mathrm{Max}(DS)=\mathrm{Max}(DS_c+DS_o+DS_q+DS_s)=0.3+0.2+0.2+0.3=1$.本文上述的权值定义和后文详述的相似度计算方法使之得以保证.

#### 2.3.1 类结构图和对象结构图的相似性度量

类结构图来自于类图,而对象结构图来自于顺序图.转换算法基于对类图的 XML 表示的分析,类图和顺序图转变为带标记的有向图(类结构图和对象结构图)后,类结构图的匹配算法与对象结构图的匹配算法是相似的,本质上都转化为子图同态检测问题.以类结构图为例:为度量相似度,首先将类结构图表示为邻接表;比较两个邻接表,查找原始设计中与反模式的类结构图相似的子图的算法如下所示:

输入:两个邻接表,一个表示反模式描述的类结构图(*adjAPD*),另一个表示原始设计的类结构图(*adjOD*);相似度阈值(*TDS*);

输出:一个邻接表集合,其中每个邻接表表示原始设计的类结构图的一个子图,并且每一个子图与反模式描述的类结构图的相似度都不小于 TDS.

(1) 根据 *adjAPD* 建立包含其所有不同节点的有序表 *seqAPD*,其顺序任意给定(一经确定后不再更改),设其元素个数为 *M*;

(2) 根据 *adjOD* 建立包含其所有不同节点的集合 *setOD*;

(3) 求出 *setOD* 的所有拥有 *M* 个元素的子集 $subSetOD=\{set_1,\ldots,set_k\}$;

(4) 对于 *subSetOD* 中任一元素 $set_i$,根据两输入邻接表中给出的节点间关系,利用试探法找出所有与 *seqAPD* 一致的形式(即:用 $set_i$ 中 *M* 个节点的每个排列组合,与 *seqAPD* 按它们各自对应的邻接表作比较).设所找的所有满足条件的有序表是 $preSeq=\{seq_1,\ldots,seq_n\}$;

(5) 对任意 $seq_i$,计算 $DS_{Ci}=\sum_{j=1}^{l}V_j$.其中 $l$ 是 $seq_i$ 中所包含的边的总数,如果 $\beta(\mathrm{edge}_j\ \mathrm{of}\ \mathrm{seq}_i)=\beta'(\mathrm{edge}_j\ \mathrm{of}\ \mathrm{seAPD})$,则 $V_j=\omega_j$;否则 $V_j=0$.如果 $DS_{Ci}\geq TDS$,则输出 $seq_i$ 的邻接表.

#### 2.3.2 质量要素的相似性度量

$$DS_q=0.2\times\left[\sum_{i=1}^{8}\left(1-\frac{|VI_i-VC_i|}{VC_i}\right)\times\frac{1}{8}\right],$$

其中 $VI_i$ 和 $VC_i$ 的含义见表 2.显然,如果对于所有 $i$,都有 $VI_i=VC_i$,则 $DS_q$ 取得其最大值 0.2.

**Table 2** Meanings of $VI_i$ and $VC_i$
表 2 $VI_i$和 $VC_i$的含义

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| VI (value of initial design segment through similarity measurement of class structure graph) | $\overline{\text{WMC}}$ | $\overline{\text{NOC}}$ | DIT | $\overline{\text{RFC}}$ | $\overline{\text{CBO}}$ | $\overline{\text{LCOM}}$ | NOAC | NRWAC |
| VC (value of design improving case) | $\overline{\text{WMC}}$ | $\overline{\text{NOC}}$ | DIT | $\overline{\text{RFC}}$ | $\overline{\text{CBO}}$ | $\overline{\text{LCOM}}$ | NOAC | NRWAC |

### 2.4 面向FACADE的事例应用

事例应用是指用检索到的事例的 PD 指明的设计方案替代原始设计中与 APD 一致的有问题的设计方案的过程.其间既可以将检索到的事例的模式描述部分不经改动地重用在给定的原始设计中,也可以经由设计专家添加、删除、更改其中的类、关系、对象和消息后再应用.我们的方法通过引进一个称作 FACADE 的封装类来隐藏微体系结构的内部信息,并提供一个外部访问接口,从而逐步完成对原始设计的改进.面向 FACADE 的事例应用算法如下所示.

输入:经检索过程后选择的具有较高相似度的事例集($S_{case}$),原始设计 $D_1$;

输出:一系列改进后的设计,其中每个设计是每次应用一个事例后的结果.

1 $i$=0;
2 While ($S_{case}\neq\varnothing$) Do
3 $D_i=D_{i-1}$; *Get one Case* $\in$ *Scase*; Let *Scase*=*Scase*−*Case*;
4 Let $S_{c1}$={*class*|*class* $\in D_{i-1}\wedge$ *class* $\notin$ *Case.APD*}
5 Let $S_{c2}$={*class*|*class* $\in$ *Case.APD*}
6 Remove all *classes* $\in S_{c2}$ from $D_i$;
7 Add a *class FACADE* to $D_i$; Add *all classes* $\in$ *Case.PD* to $D_i$;
8 Remove all *relationships* $\in$ *Case.APD* from $D_i$;
9 Add all *relationships* $\in$ *Case.PD* from $D_i$;
10 While ($S_{c1}\neq\varnothing$) Do
11 Get one $C_1 \in S_{c1}$; Let $S_{c1}= S_{c1}-C_i$;
12 While ($S_{c2}\neq\varnothing$) Do
13 Get one $C_2 \in S_{c2}$; Let $S_{c2}= S_{c2}-C_2$;
14 If ($\exists R \in$ {*Inh*,*Ini*,*Del*} in $D_{i-1}\wedge R(C_1,C_2)$) Then
15 Remove $R(C_1,C_2)$ from $D_i$; Add $R(C_1,FACADE)$ to $D_i$; EndIf;
16 If ($\exists R \in$ {*Inh*, *Ini*, *Del* } in $D_{i-1}\wedge R(C_2,C_1)$) Then
17 Remove $R(C_2,C_1)$ from $D_i$; Add $R(FACADE,C_1)$ to $D_i$; EndIf;
18 If ($\exists Agg(C_1,C_2,k) \in D_{i-1}$) Then
19 Remove $Agg(C_1,C_2,k)$ from $D_i$;*Add* $Agg(C_1,FACADE,k)$ to $D_i$; End If;
20 If ($\exists Agg(C_2,C_1,k) \in D_{i-1}$) Then
21 Remove $Agg(C_2,C_1,k)$ from $D_i$;*Add* $Agg(FACADE,C_1,k)$ to $D_i$; End If;
22 EndWhile;
23 EndWhile;
24 *OutPut*($D_i$);
25 Remove all *Case* from $S_{case}$ satisfying that $\exists$*class* $C \in$ *Case.APD* and $C \notin D_i$;
26 $i$=$i$+1;
27 EndWhile;
28 Ruturn;

### 2.5 事例的学习

基于事例推理的一个重要特点是其持续的机器学习能力.正如第 2.4 节所说,专家可以调整所选择事例的解决方案部分,使之适应特定的情况.事实上,同样可以通过手工或自动地调整事例的问题描述部分来加强推理能力.对问题描述部分的调整主要包含 3 个方面:调整类结构图中边的权值)、调整对象结构图中的边的权值(即对象消息的重要性分配)和语义约束的权值.因为对这 3 个方面的调整是相似的,我们这里仅解释对语义约束权值的重计算方法.

定义 **2.7**. 专家调整原则.每次选择一个事例后,用户可以根据自己的经验对语义约束的权值进行调整,但必须保证 $\sum_{i=1}^{n}\omega_i = 0.3$,其中 $\omega_i$ 是第 $i$ 条语义约束的权值,$n$ 是语义约束的总数.

定义 **2.8**. 令 $\omega_i|_m$ ($i$=1,...,$n$)是第 $i$ 条语义约束进行第 $m$ 次重计算和更新时在专家调整之前的权值,$\omega_i'|_m$($i$=1,…,$n$)是专家给出的调整权值,那么第 $i$ 条语义约束进行第 $m$ 次重计算后的权值赋予($\omega_i|_m$+$\omega_i'|_m$)/2.

下一轮语义约束的缺省权值根据定义 2.8 重计算和缓存.这时并不将重新计算过的事例存入事例库,这是因为专家设置和调整权值带有一定的主观性,为保证系统不出现经常性的退化,在每次设置和调整之后,对最近 10 次仅是问题描述部分权值不同的事例进行归纳和比较,取出现次数最多的事例存于事例库中.如果出现次数相等,例如这 10 个事例都只出现一次,则选择最近出现的事例存于事例库中.这种方法,我们称为投票选择原则.

定理 **2.1**. 令 $\omega_i|_m$($i$=1,...,$n$)是第 $i$ 条语义约束进行第 $m$ 次重计算后的权值,则 $\sum_{i=1}^{n}\omega_i|_m = 0.3$.

证明:使用数学归纳法.设 $m$ 是重计算的次数,$\omega_i'|_m$($i$=1,…,$n$)是第 $i$ 条语义约束在第 $m$ 次重计算时由设计专家进行调整前的权值,$\omega_i''|_m$($i$=1,…,$n$)是第 $i$ 条语义约束在第 $m$ 次重计算时由设计专家进行调整后的权值,显然有 $\omega_i|_m$=$\omega_i'|_{m+1}$.

当 $m$=1 时,亦即事例库在第 1 次重计算之前,根据定义 2.6 有 $\sum_{i=1}^{n}\omega'_i|_{m=1} = 0.3$,根据定义 2.7 有 $\sum_{i=1}^{n}\omega''_i|_{m=1} = 0.3$,于是根据定义 2.8,

$$\sum_{i=1}^{n}\omega_i|_{m=1} = \sum_{i=1}^{n}\left[\left(\omega_i'|_{m=1}+\omega''_i|_{m=1}\right)/2\right] = \frac{\sum_{i=1}^{n}\omega'_i|_{m=1}+\sum_{i=1}^{n}\omega''_i|_{m=1}}{2} = \frac{0.3+0.3}{2} = 0.3.$$

假设当 $m$=$k$–1 时命题成立,即 $\sum_{i=1}^{n}\omega_i|_{m=k-1} = 0.3$,那么当 $m$=$k$ 时,

$$\sum_{i=1}^{n}\omega_i|_{m=k} = \sum_{i=1}^{n}\left[\left(\omega_i'|_{m=k}+\omega''_i|_{m=k}\right)/2\right] = \frac{\sum_{i=1}^{n}\omega'_i|_{m=k}+\sum_{i=1}^{n}\omega''_i|_{m=k}}{2} = \frac{\sum_{i=1}^{n}\omega'_i|_{m=k-1}+\sum_{i=1}^{n}\omega''_i|_{m=k}}{2} = \frac{0.3+0.3}{2} = 0.3.$$

为避免事例之间的矛盾性,在专家设置和调整权植之后和调整后的事例重新存储到事例库之前,通常进行知识的一致性检查.具体地,根据修改后的事例问题描述部分,按照第 2.3 节所述的检索方法检索事例库.如果检索结果存在,并且检索结果的解决方案部分与修改后的事例问题解决部分不同,则按照投票选择原则选取相应的事例替代事例库中原有的事例,从而解决了不一致知识的使用问题.

## 3 基于 CBR 的设计改进工具 CBDIT(CBR-based design improving tools)

图 3 显示了 CBDIT 的体系结构.

CBDIT 包含 4 个工具和 3 个库:UML 图形编辑器、设计抽取工具、数据库和知识库管理工具、CBR 推理引擎、事例库、语义约束规则库和同义词库.

UML 图形编辑器用于可视地以鼠标和键盘绘制类图和顺序图.该工具用 XML 存储显示信息和内部设计信息(静态结构信息和动态行为信息).静态结构和对应的 XML 表示如图 2 所示.

设计抽取工具用于从文档不完整的遗产系统抽取设计信息.采用逆向工程的通用模型,设计抽取工具,抽取源码的静态和动态信息.静态信息通过对头文件和实现文件的静态分析得到.例如类、属性、方法和继承关系

可以直接从头文件中抽取,作为伪属性、形参和方法返回类型,部分聚集和委托关系也可以参考头文件得到.而方法定义中局部变量的类型则可以指明剩余的委托关系,该信息可以从实现文件中得到.动态信息在跟踪运行时刻的行为过程中收集,补充了静态信息.该工具跟踪系统事件,如对象创建、销毁、方法进入、方法出口等,并记录方法的参与者来构造消息列表,从而最终生成顺序图.

数据库和知识库管理工具用于创建、删除、修改和查询存储在数据库和知识库中的数据和知识.它为用户管理数据库和知识库提供了图形用户界面.

CBR 推理引擎是 CBDIT 的核心,其推理机制已经在前文中详述.

事例库、语义约束规则库和同义词库分别存储改进事例、语义约束和同义词.其中同义词库对于事例库的索引和语义约束的判定具有重要意义.

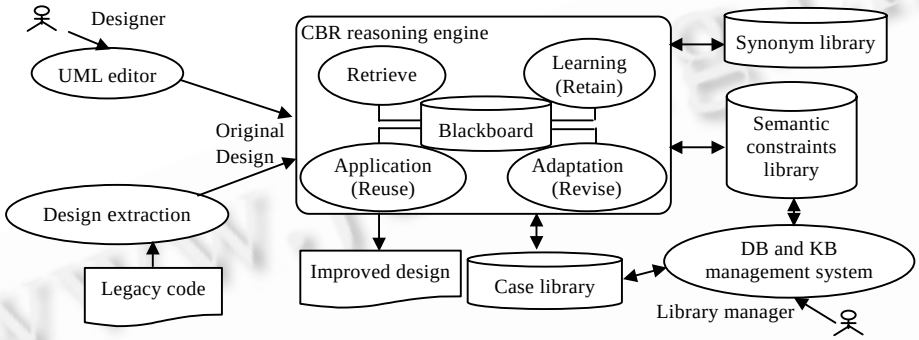



Fig.3 Architecture of CBDIT

图 3 CBDIT 的体系结构

## 4 相关工作

近年来,一些软件设计改进和利用 CBR 技术进行设计模式应用的研究正成为一个新的热点.

Gomes[8,9]等人提出一种利用 CBR 实现软件设计模式自动化的方法,并开发了相应的支撑系统 REBUILDER.与我们的工作相比,他们的研究不足在于:他们提出的事例仅描述了模式应用可能的上下文,而没有涉及微体系结构反模式及其重构方案.因此,它无法保证模式应用的必要性;同时,在处理模式应用问题时,他们为每个模式定义一个模式应用算法,这使学习和转化成为部分的和不完整的.

Correa[10]提出了一种基于启发式规则、设计模式和反模式的面向对象设计经验重用方法,并开发了辅助工具 OOPDTool,但他们采用元模型谓词的知识表示和基于规则的推理机识别设计结构.

Gronback[11]总结了在 Borland Together 控制中心用审查、度量和重构对软件再建模的方法和经验.他们的方法是以质量要素的自动化度量结果和审查结果为依据,但仅提供了基于对话的手工修改方式.

Daengdej[12]尝试将 CBR 用于设计重用.但他们的方法没有应用设计模式或识别反模式,只是根据领域相关的关键字在设计结果数据库中查找相似的实体关系图.这种方法的局限性在于:用关系名称和实体名称为关键字所检索出来的设计,往往与期望得到的设计在语意上有较大的偏差,从而降低了识别的准确率.

## 5 结论和未来工作

本文提出了一种基于 CBR 和微体系结构反模式的设计改进方法,形式化地定义了在微体系结构层中不灵活的设计结构和相应的重构方案的描述方法,研究了它们在事例库中的组织和索引机制.根据基于事例推理技术的 4R 过程,本文提出了类图、顺序图、质量要素和语义约束的相似性度量方法,描述了在给定设计中识别反模式及其实例,并在此基础上实现了进行替代的几种算法、设计改进支撑系统——CBDIT 的体系结构的设计.

进一步的研究工作有:进一步改进方法和工具,准确识别编程人员在实现过程中所使用的微体系结构反模式的某种变体;提供更高效的、充分利用领域知识进行快速剪枝的多阶段图形检索算法.

**References**:

[1] Gamma E, Helm R, Johnson R, Vlissides J, Design Patterns: Elements of Object-Oriented Software Architecture. New York: Addison-Wesley, 1995.

[2] Brown WH, Malveau RC, “Skip” McCormick III HW, Mowbray TJ. Antipatterns: Refactoring Software, Architectures, and Projects in Crisis. New York: Wiley Computer Publishing, 1998.

[3] Sanders KE, Kettler BP, Hendler JA. The case for graph-structured representations. In: Leake DB, Plaza R, eds. Proc. of the 2nd Int’l Conf. on Case-Based Reasoning Research and Development. Berlin: Springer-Verlag, 1997. 245−254.

[4] Plaza E. Cases as terms: A feature term approach to the structured representation of cases. In: Veloso MM, Aamodt A, eds. Proc. of the 1st Int’l Conf. on Case-Based Reasoning Research and Development. Berlin: Springer-Verlag, 1995. 265−276.

[5] Chidamber SR, Kemerer CF. A metrics suite for object-oriented design. IEEE Trans. on Software Engineering, 1994,20(6): 476−493.

[6] Silverston L. The Data Model Resource Book Revised Edition Volume 2: A Library of Universal Data Models by Industry Types. 2nd ed. New York: John Wiley & Sons, 2001.

[7] Booch G, Jacobson I, Rumbaugh J. The unified moddeling language for object oriented development—UML Semantics—version 1.1. 1997. http://www.omg.org

[8] Gomes P, Pereira FC, Paiva P, Seco N, Carreiro P, Ferreira J, Bento C. Using CBR for automation of software design patterns. In: Craw S, Preece AD, eds. Proc. of the European Conf. on Case-Based Reasoning (ECCBR 2002). Berlin: Springer-Verlag, 2002. 534−548.

[9] Gomes P, Pereira FC, Paiva P, Seco N, Carreiro P, Ferreira J, Bento C. Selection and reuse of software design patterns using CBR and wordNet. In: Proc. of the 15th Int’l Conf. on Software Engineering and Knowledge Engineering (SEKE 2003). San Francisco, 2003. 289−296. http://www.ksi.edu/seke/seke03.html

[10] Correa AL, Werner CML, Zaverucha G. Object oriented design expertise reuse: An approach based on heuristics, design patterns and anti-patterns. 2000. http://citeseer.ist.psu.edu/439618.html

[11] Gronback RC. Software remodeling: Improving design and implementation quality. A Borland White Paper, 2003. http://www.togethersoft.com/events/webinars/archive/ software_remodeling_whitepaper.pdf

[12] Daengdej J, Moemeng P, Charoenvikrom S. Toward the use of case-based reasoning for design reuse. 2002. http://citeseer.ist.psu.edu/534321.html

冯铁(1972－),男,吉林长春人,博士生,副教授,主要研究领域为软件复用,软件体系结构,自动化软件工程方法和工具.

王洪媛(1974－),女,讲师,主要研究领域为需求工程,建模技术.

张家晨(1969－),男,博士,教授,CCF 高级会员,主要研究领域为软件演化和重构.

金淳兆(1937－),男,教授,博士生导师,主要研究领域为软件工程.