模型驱动环境下模型演化的形式化研究

孙为军[1,2] 李师贤[1] 严玉清[1,3]
(中山大学计算机科学系 广州 510275)[1] (广东工业大学计算机学院 广州 510006)[2]
(广东外语外贸大学思科信息学院 广州 510006)[3]

摘 要 在模型驱动开发中,模型演化由一系列复杂的变化活动组成,模型的变化可以分为直接施加在模型元素上的增加、删除、更改等基本演化操作以及这些基本演化操作的组合。基于模型驱动体系结构,给出了模型和模型变化的形式化定义。模型的变化以模型差异来描述,在模型差异的基础上,研究了模型的合并、逆和组合运算。

关键词 模型驱动体系结构,模型演化,模型差异,元模型
中图法分类号 TP311.5 **文献标识码** A

Study on the Formalization of Model Evolution with Model Driven Architecture

SUN Wei-jun[1,2] LI Shi-xian[1] YAN Yu-qing[1,3]
(Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China)[1]
(Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, China)[2]
(Cisco School of Informatics, Guangdong University of Foreign Studies, Guangzhou 510006, China)[3]

Abstract Model evolution involves a series of complex change activities. Based on model driven architecture, the relevant model evolution conceptions were formally defined to accurately describe model and model changes. In MDE, model changes can be classified as the primitive operations and composition operations. The primitive operations are identified as addition, deletion and modification operation and can be imposed on the model element individually. Merge and union operations are composition operations based on simple operations.

Keywords Model driven architecture, Model evolution, Model difference, Metamodel

1 引言

软件演化已成为软件生命周期中最重要的形态之一。随着运行环境以及需求的不断变化,软件(模型)也需要不停地演化以满足新的需求[1]。软件演化由一系列复杂的变化活动组成,软件演化的过程也就是通过修改软件的组成成分以适应软件变化的过程。研究表明,80%的软件开发成本用于软件修改,以满足不断变化的软件需求。这些变化使得软件演化的控制变得越来越困难。如果不对软件变化给予及时地处理,将会使软件系统变得更为复杂与难以维护。

MDA方法以模型为中心,通过对业务逻辑建立抽象模型,经过模型转换和逐步精化,自动生成最终软件产品。模型构造、模型转换和精化是模型驱动体系结构的核心[2]。在模型驱动开发中,模型、元模型和模型转换规则都可能发生变化。模型的变化用模型差异来表示,模型演化操作的序列构成模型差异,对模型驱动软件演化的研究可以归结为对模型差异的研究。

为了支持模型驱动开发中的模型演化,需要精确地定义和描述模型和模型的变化,为此提出了一套有关模型演化概念的形式化定义。用精确的符号来表示对模型和模型变化的形式化描述,其具有精确数学语义,能够进行推理和演算,对模型的版本管理、变化管理以及软件开发的自动化都有非常重要的意义。

2 相关工作

在模型演化领域,已经有了很多关于版本、差异、比较、合并和组合的研究。Alanen[3]给出了获取模型差异的算法,研究了怎样把并行的模型更改合并成一个模型,给出了3个关于模型的差异、合并和组合的通用算法,这些算法中的模型元素依赖于对象通用唯一标识符(Universally Unique Identifier, UUID),适用于任何基于元对象设施(MOF)的建模语言。但是Alanen并没有给出模型差异的形式化描述。Treude[4]提出了一个不依赖于UUID的模型比较算法。Oliveira[5]给出了一个UML的版本控制系统来为与UML一起工作的CASE工具提供配置管理支持。该系统满足基本版本控制的需要,支持UML模型的差异计算、修改和合并,但他们的方法仅限于UML模型,它允许用户定义比较单元和版本单元来满足他们的具体需求。MetaDiff[6]为模型比较提供了一个框架。该框架本身没有为比较和合并模型提供算法,旨在测试基于MOF的建模语言的不同算法。Toulme[7]通过

到稿日期:2011-10-20 返修日期:2011-12-10 本文受国家自然科学基金(60774095),广东省自然科学基金(9451009001002777)资助。
孙为军(1975—),男,博士生,讲师,主要研究方向为软件工程、模型驱动体系结构、软件演化,E-mail:gdutswj@gdut.edu.cn;李师贤(1944—),男,教授,博士生导师,主要研究方向为软件工程、形式语义学;严玉清(1963—),女,博士生,副教授,主要研究方向为软件工程、需求管理。

为 EMF 模型提供一个实用工具来进行模型的比较，相对于Oliveira 所做的工作，该工具不适用于模型合并。该工具不依赖于 UUID，而是使用语义来匹配模型元素。Altmanninger[8] 对模型合并中的冲突检测模型进行了语法与语义的分析，以便进行更精确的冲突检测，并且确定冲突的原因。Brunet 等[9] 提出了一套模型管理操作（合并、匹配、差异和分解等），并研究了每种操作的代数性质。

3 模型演化概述

MDA 方法中，模型转换可以分成垂直模型转换[10,11] 和水平模型转换[12,13]。垂直模型转换是将一个源模型转换到不在同一抽象级别上的目标模型，如平台无关模型 PIM 到平台相关模型 PSM 及 PSM 到代码的转换。水平模型是将一个源模型转换成同一抽象级别上的另一个目标模型。水平转换主要用于模型演化或模型重构。

MDA 以模型为中心，为了描述模型的变化，给定了同一抽象级别上的两个模型 M 和 M'。M 是变化前的模型，即源模型；M' 是变化后的模型，即目标模型。M 经过一定的变化得到目标模型 M'。这些变化称为模型差异，记为 $\Delta M = M' - M$。这个变化过程称为模型演化过程，模型演化过程包含了一系列变化操作，这些变化操作包括增加、删除或更改模型中的一个或多个元素。通过合适地组织变化操作，将这些变化操作依次应用于源模型，就可以得到目标模型。模型演化的变化操作序列具有可再现性，通过对初始源模型重复应用这些操作，目标模型被重复创建。为了表述方便，本文采用符号 $\otimes$ 作为模型演化算子，$M \otimes \Delta M$ 表示在模型上 M 上实施变化操作 ΔM 后所得的新的模型。

在模型演化中，通过模型转换规则，对源模型的变化必须被映射到目标模型。模型演化的目标，是描述一个模型 M_1 的一个变化模型 ΔM_1，然后通过模型转换得到依赖于 M_1 的其他模型的变化。例如，模型 M_1 可以通过模型转换 T 得到模型 M_2，即 $M_2 = T(M_1)$。如果 M_1 变化为 ΔM_1，ΔM_1 经过模型转换生成 ΔM_2，再利用 M_2 和 ΔM_2 生成变化后的目标模型，这个目标模型等价于将 ΔM_1 应用到 M_1 上后再进行转换得到的目标模型，即 $M_2 \otimes \Delta M_2 = T(M_1 \otimes \Delta M_1)$。通过 ΔM_2 计算目标模型的代价要比通过计算 $M_1 \otimes \Delta M_1$ 再进行转换的代价小。

图 1 表示了模型差异在模型驱动框架中的应用。模型差异处理模块根据源模型 M_1 和源模型变化 ΔM_1，通过转换规则生成目标模型 M_2 和 M_2 以及模型变化 ΔM_2。

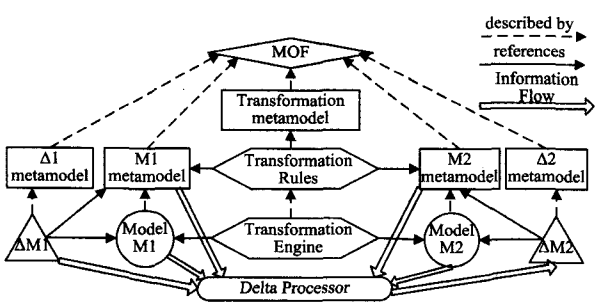

图 1 模型演化处理

4 模型演化相关定义

4.1 元模型与模型的定义

定义 1 一个元模型（metamodel）定义为一个二元组 $MM=(N,E)$，一个元模型被用于定义某一类模型所必须遵守的原则和规范。其中：

（1）N 是节点的集合，代表元模型中的模型元素。$N=\{n_1,n_2,\cdots,n_n\}$，$\forall n \in N, n=\langle id, nodetype, value\rangle$。式中，$id$ 为模型元素的标识符；$value$ 代表模型元素的值，可以为 Null；一般的模型元素都是有类型的，$nodetype$ 为模型元素的类型，如果一个模型元素没有指定类型，则默认其类型为 Null。

（2）E 是有向边的集合，代表元素之间的连接。$E=N\times N\times\{edgetype\}$。两个 N 表示元素连接的两个端点。$e=\{e_1,e_2,\cdots,e_n\}$，$\forall e \in E, e=\langle n_s, edgetype, n_t\rangle, s,t \in \{1\cdots n\}$。式中，$n_s$ 代表源模型元素；n_t 代表目标模型元素；n_s，n_t 可以为同一元素；$edgetype$ 代表连接的类型，有 4 种基本的连接类型：泛化关系、包含关系、关联关系、聚合或组合关系。

定义 2 一个模型（Model）定义为一个元组 $M=(N,E,MM,\tau)$，当满足一系列条件时，一个模型符合（Conform to）某个元模型，此时，称这个模型为其元模型的“实例”：

（1）MM 是一个元模型，用来定义模型 M。

（2）N 是节点的集合，代表模型 M 中的模型元素，这些节点是元模型中模型元素的实例。

（3）E 是边的集合，$E=N\times N$，代表模型中的连接，这些边是元模型中的边的实例。

（4）τ 是函数类型，$(N\rightarrow N_{MM})\cup(E\rightarrow E_{MM})$。$\tau$ 包含了模型中模型元素和连接的类型信息，即 E 为 E_{MM} 的实例，N 为 N_{MM} 的实例。

定义 3 模型符合（Model Conformance） 给定一个元模型 $MM=(MMN,MME)$，一个模型 $M=(N,E,MM,\tau)$ 是符合元模型 MM 的，或者说模型 M 是良好定义的，当且仅当满足下列 3 个条件：

（1）$\forall n_i \in N, \exists mn_k \in MMN, \tau(n_i)=mn_k$；

（2）$\forall e_i \in E, \exists me_k \in MME, \tau(e_i)=me_k$；

（3）$\forall e \in E, \tau(fst(e))=fst(\tau(e)) \wedge \tau(snd(e))=snd(\tau(e))$。

条件（1）意味着模型元素 n_i 是元模型 mn_k 的一个实例。条件（2）意味着一个模型 M 中的一个连接 e_i 是一个元模型 MM 中的一个连接 me_k 的实例。条件（3）对应连接两端的元素类型的兼容性，这个条件是为了维持元模型中模型元素间的关系，其中 $fst(e)$ 和 $snd(e)$ 分别表示连接的 e 源端节点和目的端节点。

4.2 模型演化操作

模型演化操作的描述包括两方面的主要内容，一个是演化操作的名称和参数，其次是模型演化操作的前提条件和后置条件描述。

定义 4 一个模型演化操作是一个五元组 $op=\langle Name, I, O, Pre, Post\rangle$，其中：

（1）$Name$ 为演化操作的名称；

（2）I 为演化操作的输入参数列表，包含了一些变量；

（3）O 为演化操作的输出参数列表，包含了一些变量；

（4）Pre 为演化操作得以实施的前提条件，是一个逻辑表

达式，调用演化操作前必须满足；

(5) *Post* 为演化操作实施后必须满足的后置条件，是一个逻辑表达式。后置条件描述了演化操作调用后模型状态的变化。

从模型演化操作的定义可以看出，模型演化操作是源模型到目标模型之间的映射。演化操作的语法表示包括了输入和输出参数，操作的语义则规定了调用这个操作前必须满足的前提条件和调用后必须满足的后置条件。前提条件 *Pre* 用于查询需要变更的模型成分，并确定对模型成分的变更是否合法。后置条件 *Post* 包含了肯定谓词和否定谓词，分别表示演化操作后必须存在的模型成分和不再存在的模型成分。前提条件和后置条件的描述需要用到一些重要的谓词项。例如，谓词 exists 用来描述模型成分是否存在。

根据模型演化操作粒度的不同，演化操作可以划分为基本演化操作(Elementary Evolution Operation)和组合演化操作(Composite Evolution Operation)。基本演化操作通过组合，可以得到复杂的组合模型操作。

对于一个演化操作而言，其前提条件和后置条件相对比较固定，它不依赖于特定的模型和演化参数。描述模型演化操作时，通常可以忽略演化的前提条件和后置条件，而只采用 name(Input,Output)进行表示，其中参数 Input 和 Output 为输入参数和输出参数的列表。

定义 5 基本演化操作是对模型中单个模型元素或特性的演化操作，这些演化操作对模型的更改不可分割，不能做进一步分解。

基本演化操作是原子操作，可以单独作用在模型元素上。与模型变化有关的基本演化操作分为 3 类：增加(Addition)、删除(Deletion)和更改(Modification)，如图 2 所示。

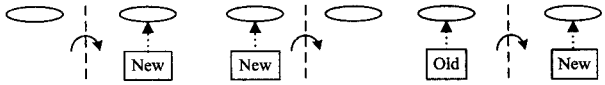

图 2 3 类基本的演化操作

以模型和元模型中定义的基本模型元素为基础，抽象出下列 10 种增加操作、删除操作和修改操作，如表 1 所列。

表 1 基本演化操作

序号	基本演化操作	操作语义
1	addNode(nl,nt)	增加节点
2	delNode(nl,nt)	删除节点
3	addEdge(el,et,sl,st,tl,tt)	增加节点
4	delEdge(el,et,sl,st,tl,tt)	增加边
5	addNodeAttr(nl,nt,at,an,av)	增加节点属性
6	delNodeAttr(nl,nt,at,an)	删除节点属性
7	setNodeAttr(nl,nt,at,an,v_0,v_1)	修改节点属性
8	addEdgeAttr(el,et,at,an,av)	增加边属性
9	delEdgeAttr(el,et,at,an)	删除边属性
10	setEdgeAttr(el,et,at,an,v_0,v_1)	修改边属性

表 2 给出了基本演化操作 addNode 的语法、语义、前提条件和后置条件。

由于篇幅限制，表 1 忽略了基本演化操作的前提条件和后置条件。需要注意的是，基本演化操作有两个特点：每一个操作都是完整的，可以单独地作用在模型上；其次，每一个操作都对应一个撤销的逆操作，如 addNode 操作和 delNode 操作是互逆的。

表 2 基本模型演化 AddNode 操作详细描述

操作名称	addNode
语法	addNode(nodelabel,nodetype)
语义	创建一个新的节点，该节点的类型为 nodetype
前提条件	¬exists(node(nodelabel,nodetype)
后置条件	exists(node(nodelabel,nodetype)

定理 1 对于模型的核心元素和特性而言，表 1 所列的基本演化操作集是完备的。即对于任意的两个模型 M_1 和 M_2，存在一个由表 1 所列的基本变化操作组成的有限变化序列 $\Delta M=[op_1,op_2,\cdots,op_n]$，使得 $M_2=M_1\otimes\Delta M$。

证明：对于任意的两个模型 M_i 和 M_2，核心的元素和特性都是用节点和边来表示的，可以通过如下两个步骤将 M_i 修改为 M_2：

(1)通过如下步骤可以将模型 M_1 中所有的节点和边清空：假定模型图中存在节点和边，首先可以通过多次利用 delEdgeAttr和 delEdge 将所有的边去掉，将该模型编程为一个不含任何边的模型。然后，通过多次利用 delNodeAttr 和 delNode 将模型中的所有节点删除，得到一个空的模型。

(2)显然，在与步骤(1)互逆的演化操作中，可采用有限的演化操作构建模型 M_2。

由于步骤(1)能够通过有限的基本演化操作组合，将模型 M_1 中所有的核心元素和特性删除，模型 M_2 也能够通过步骤(2)进行构建，因此表 1 所列的基本演化操作集合对于模型核心元素和特性的修改而言是完备的。

定理 1 保证了针对模型核心元素和特性的任意变化需求都是可以表示的，这也是模型演化形式化表示的理论基础。

定义 6 组合演化操作是由有限个基本演化操作或者其他组合演化操作组合而成的模型演化操作。

常用的组合模型演化操作包括模型元素的分解和合并、模型元素的移动等。组合演化操作不存在一个完备集，可以根据模型演化的需要定义新的组合演化操作。组合演化操作的语义由组成它的基本演化操作的语义确定。表 3 列举了部分组合演化操作。

表 3 组合演化操作(部分)

序号	组合演化操作	操作语义
1	moveUpNode	提升节点的继承层次
2	moveDownNode	降低节点的继承层次
3	copyNode	通过拷贝来构件一个新的节点
4	delNodeTree	删除节点(包括所有子节点和边)
5	splitNode	将一个节点分解为多个节点
6	mergeNode	将多个节点合并为一个节点
7	moveNodeAttribute	将一个节点属性迁移到另一个节点

定义 7 对一个模型 M 而言，一个模型演化操作 $op=\langle name,I,O,Pre,Post\rangle$ 是可调用的，当且仅当演化操作 op 的前提条件 Pre 在模型 M 的状态 I_M 下成立，记为 $\vdash_M Pre$。

演化操作的实例由演化操作 op 和操作对象 M 共同组成，一个演化操作的调用实例表示为 $i=(op,Z_M)$，其中 $op=\langle name,I,O,Pre,Post\rangle$，$Z_M$ 是根据模型 M 对操作中每一个变量的赋值。

定义 8 给定两个演化操作 $op_1=(name_1,I_1,O_1,pre_1,post_1)$ 和 $op_2=(name_2,I_2,O_2,pre_2,post_2)$，如果 $\exists x,x\in O_1\wedge x\in I_2$，则演化操作 op_1 和 op_2 是相关的，并且有 $op_1\rightarrow op_2$，表示 op_1 和 op_2 之间存在一个演化路径。

一个演化操作的相关性说明一个演化操作的输出将被后

继的基本演化操作使用，它们之间具有依赖关系。

定理 2 对模型 M 的模型演化操作序列$[op_1, op_2]$而言，当模型演化操作 op_2 依赖于模型演化操作 op_1 时，满足 $(Post_1(I_M)\backslash I_M)\cap Pre_2\neq\emptyset$。

证明：模型演化操作 op_2 依赖于软件演化操作 op_1 时，模型演化操作 op_2 的前提条件 Pre_2 只有在实施模型演化操作 op_1 后才能成立。这表明，模型演化操作 op_1 对模型 M 实施修改后的模型状态 $Post_1(I_M)$ 使模型演化操作 op_2 的前提条件 Pre_2 为真，但 Pre_2 在 I_M 下为假。从而，在 $Post_1(I_M)$ 中删除掉与 I_M 相同的状态(即 $Post_1(I_M)\backslash I_M$)后，与 Pre_2 的交集不为空[14]。

4.3 模型差异

模型的变化用模型差异 Δ 来描述。模型差异 Δ 定义为一系列对模型中元素的演化操作，这些演化操作包括增加、删除或更改模型中的一个或多个元素。将模型差异 Δ 应用于 M_1，就是按照一定顺序执行 Δ 中的操作，从而得到 M_2。显然，这种方法直观地描述了模型演化的过程。

定义 9(模型差异，Model Difference) 给定两个模型 $M_1=(N_1, E_1, MM_1, \tau)$，$M_2=(N_2, E_2, MM_2, \tau)$。$M_1$ 经过有限个演化操作得到 M_2，则 M_2 和 M_1 的模型差异记为 $\Delta=M_2-M_1=[op_1, op_2, \cdots, op_n]$。

模型差异 Δ 定义为一个模型演化操作的有序集合 Δ。[] 表示顺序性，即其中的演化操作必须按照 $op_1\rightarrow op_2\rightarrow\cdots\rightarrow op_n$ 的顺序执行。

根据模型差异的定义，模型差异就是引起模型变化的模型演化操作序列。对于模型演化操作而言，模型操作序列将构成一个偏序依赖关系，即任意两个模型演化操作实施时间先后依赖关系满足自反性、反对称性和传递性。模型演化操作的偏序依赖关系表明模型演化操作的调用顺序是至关重要的。

根据模型演化操作和演化对象的特点，可以对模型差异实施合并运算、逆运算和组合运算。

(1)合并运算(Merge)

将模型差异合并到模型上是指将模型差异中包含的演化操作应用到模型上。给定一个模型 M 和一个模型差异 Δ，合并运算表示为 $M+\Delta$ 或者 $\Delta(M)$，这意味着将 Δ 中的演化操作按照给定的顺序施加在模型上。

(2)逆运算

给定两个模型 M_1 和 M_2，以及模型差异 Δ，有 $M_1+\Delta=M_2$，可以计算逆$\overline{\Delta}$，满足 $M_2+\overline{\Delta}=M_1$。差异运算的逆运算是一个简单的过程，只需要逆向排列 $\Delta=[op_1, op_2, \cdots, op_n]$ 的演化操作序列，并对每一个演化操作用其逆操作取代就可以得到模型差异的逆$\overline{\Delta}$。

(3)组合运算

假设有两个模型差异 Δ_1 和 Δ_2，要被依次合并到模型 M 上，即 $M_U=\Delta_1(\Delta_2(M))$，记 $\Delta'=\Delta_1\oplus\Delta_2$，$M_U=\Delta_1(\Delta_2(M))=\Delta'(M)$，$M_U$ 为 Δ_1 和 Δ_2 在 M 上的组合运算的结果。

4.4 组合运算的冲突解决

实施两个模型差异中的组合运算的目的是选择各个模型差异中合适的演化操作并规划操作执行的先后顺序。模型差异组合运算的关键是各种不同的演化操作的优化和冲突解决办法。

定义 10(演化操作冲突，Conflict) 两个模型演化操作 op_1 和 op_2 的执行相互冲突，当满足下列条件时：

(1)op_1 和 op_2 存在互逆关系：$op_1=\neg op_2$；

(2)op_1 删除了 op_2 的前提条件，即

$x\in Post_1\wedge x\in Pre_2$；

(3)op_1 的执行结果不满足 op_2 的前提条件，即

$x\in Post_1\wedge x\notin Pre_2$；

(4)op_1 和 op_2 的后置条件不一致，即

$x\in Post_1\wedge x\notin Post_2$；

(5)op_1 的变化包含了 op_2 的变化，即

$Pre_2\subseteq Pre_1\wedge Post_2\subseteq Post_1$。

一个演化操作序列$[op_1, \cdots, op_n]$是冲突的，当存在两个演化操作 op_i 和 $op_j(i\neq j)$ 是冲突的。在模型差异 Δ_1 和 Δ_2 的组合运算中，可能发生多种冲突[15]。

条件(1)对应的冲突称为互逆冲突，如果参与组合的演化操作 op_1 和 op_2 是互逆关系，那么 op_1 和 op_2 的执行效果互相抵消，因此从组合效果来看其是冗余的。解决办法是忽略这两个操作。

条件(2)－(4)对应的冲突称为覆盖冲突，如果 op_1 和 op_2 之间存在覆盖关系，那么 op_2 从效果上来说是冗余的，因为演化操作 op_1 的执行效果将导致 op_2 无法执行。元素删除可能引发覆盖冲突，最糟糕的冲突情况是，当 Δ_1 包含删除元素 e 的操作，而且 Δ_2 包含修改元素 e 的操作。那么解决这种冲突的办法是忽略这种修改，并且删掉元素。

条件(5)对应的冲突为触发冲突，操作 op_1 包含操作 op_2，op_1 的执行会触发 op_2，因此 op_2 是冗余的。这种冲突的一种解决办法就是忽略被包含的操作。

结束语 本文给出了模型驱动环境下模型演化相关概念的形式化定义，包括模型、元模型、模型符合和模型差异等。模型差异的形式化定义着重于从演化过程描述模型变化，在模型差异运算的基础上，研究了模型差异的合并、逆运算、组合运算等。

下一步的研究工作是从应用实际出发对模型变化描述进行全面分析和归纳，从而提高模型变化的语义表述能力、特性保持和一致性判定能力，使其更加准确、有效地反映并支持工程化的模型驱动开发。

参 考 文 献

[1] 刘辉，麻志毅，邵维忠. 模型转换中特性保持的描述与验证[J]. 软件学报，2007，18(10)：2369-2379

[2] 刘静，何积丰，缪淮扣. 模型驱动架构中模型构造与集成策略[J]. 软件学报，2006(06)

[3] Alanen M，Porres I. Difference and Union of Models[M]. Lecture Notes in Computer Science：Springer，2003

[4] Treude C，Berlik S，Wenzel S，et al. Difference computation of large models[C]// ESEC-FSE '07. New York，NY，USA：ACM，2007

[5] Oliveira H，Murta L，Werner C A U. Odyssey-VCS：a flexible version control system for UML model elements[C]// SCM'05. New York，NY，USA：ACM，2005

[6] Kofman M，Perjons E. MetaDiff-a Model Comparison Framework[EB/OL]. http：//metadiff. sourceforge. net/docs/metadiff. pdf

(下转第 150 页)

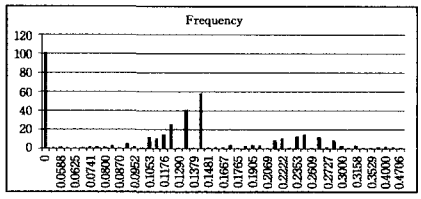

图 2　最长相同子串

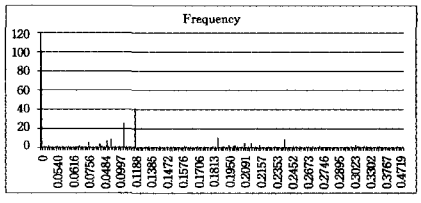

图 3　ESSK

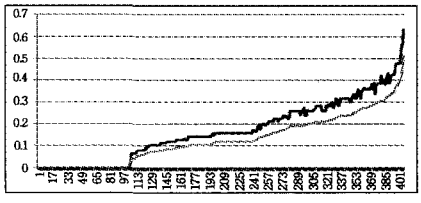

图 4　SSK(k=1)&ESSK

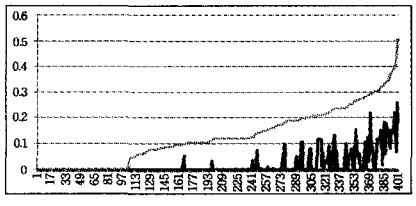

图 5　SSK(k=2)&ESSK

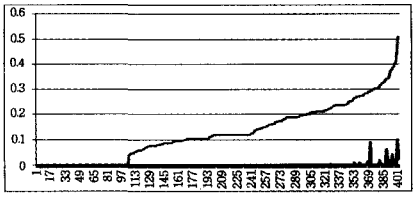

图 6　SSK(k=3)&ESSK

图 4—图 6 比较了由 SSK 和 ESSK 计算得到的相似度的值。横轴是点击流的 id,纵轴是相似度。浅色线代表 ESSK,深色线代表 SSK。点击流的 id 根据相似度的值从小到大排列,以便观察实验结果。从图 4 可以看出,当 k=1 时,SSK 计算得到的相似度值比 ESSK 大。从图 4 和图 5 可以看出,当 k=2 和 k=3 时,SSK 计算得到的相似度值比 ESSK 小。实验结果表明,ESSK 的相似度值稳定适中,更适合计算点击流相似度。

结束语　本文对 SSK 进行扩展,提出一种新的计算点击流相似度的方法 ESSK,并提出一种高效计算 ESSK 的算法。本文通过理论分析和实验结果表明,ESSK 比传统方法("编辑距离"、"最长相同子串"和 SSK)更适合计算点击流相似度。

参 考 文 献

[1] Srivastava J, Cooley R, Deshpande M, et al. Web usage mining: Discovery and applications of usage patterns form Web data[C]// SIGKDD Explorations. 2000

[2] 郭岩,白硕,于满泉. Web 使用信息挖掘综述[J]. 计算机科学,2005,32(1):1-7

[3] Cooley R W. Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data[D]. Minnesota, USA: University of Minnesota, 2000

[4] 张波,巫莉莉,周敏. 基于 Web 使用挖掘的用户行为分析[J]. 计算机科学,2006,33(8):213-214

[5] 马超,沈微. 基于闭合有间隔频繁子序列的点击流聚类[J]. 计算机工程,2010(23):72-75

[6] Lodhi H, Saunders C, Shawe-Taylor J, et al. Text Classification Using String Kernels[J]. The Journal of Machine Learning Research, 2002(2):419-444

[7] Koch D. Study of the discrepancy between client-and server side logging of clickstreams[M]. Stockholm: Royal Technical Institute, 2006

[8] Kothari R, Mittal P, Jain V, et al. On Using Page Cooccurrences for Computing Clickstream Similarity[C]// Proceedings of SIAM International Conference on Data Mining. 2003

[9] Banerjee A, Ghosh J. Clickstream clustering using weighted longest common subsequences[C]// Proceedings of the Web Mining Workshop at the SIAM Conference on Data Mining. 2001

[10] Hay B, Wets G, Vanhoof K. Clustering navigation patterns on a website using a sequence alignment method[C]// Proceedings 17th Conference on Artificial Intelligence, Intelligent Techniques for Web Personalization. 2001

[11] Wang W, Zaiane O R. Clustering Web sessions by sequence alignment[C]// Proceedings of DEXA. 2002

[12] Gunduz S, Ozsu M T. A Web Page Prediction Model Based on ClickStream Tree Representation of User Behavior[C]// Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003

[13] NASA Kennedy Space Center Log[EB/OL]. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

(上接第 119 页)

[7] Brun C, Musset J, Toulme A. EMF Compare[EB/OL]. http://wiki.eclipse.org/index.php/EMF_Compare, 2007

[8] Altmanninger K. Models in conflict-Towards a semantically enhanced version control system for models[M]. Nashville. TN, United states, 2008

[9] Brunet G, Chechik M, Easterbrook S, et al. A manifesto for model merging[C]//GaMMa '06. New York, NY, USA: ACM, 2006

[10] 尹剑飞,郭荷清,彭新一. 基于模型转换实现行为协议的研究[J]. 计算机工程,2005,31(1):31-32

[11] Yin Jian-fei, H G X P. Pattern Semantic Link: A Reusable Pattern Representation in MDA Context[C]// ICDCIT 2004. LNCS 3347, 2004:310-317

[12] Peltier J, B G G M. MTRANS: A General Framework Based on XSLT for Model Transformations[C]// Proceedings of the Workshop on Transformations in UML. 2001

[13] Gray J, Z Y L J. Model-Driven Program Transformation of a Large Avionics Framework[C]// GPCE 2004. LNCS 3286, 2004:361-378

[14] 任胜兵. 基于图变换的可视化层次用例建模及演化方法研究[D]. 长沙:中南大学,2007

[15] 鲍爱华. 语义 Web 环境下组合服务演化方法及其关键技术研究[D]. 长沙:国防科学技术大学,2009