基于GRoundTram的软件产品线设计模型的同步方法

邱连启 沈立炜 赵文耘
(复旦大学计算机科学技术学院 上海 201203)

摘 要 领域设计模型与应用系统设计模型是软件产品线开发与定制阶段的重要产物。在产品线的生命周期中,为了保证这两个模型之间的一致性,需要采用自动化或半自动化的手段实现模型之间的同步。针对该问题,提出了一种基于GRoundTram的软件产品线设计模型的同步方法,称为SPLSync-GRoundTram。该方法将领域设计模型与应用系统设计模型之间的同步问题转换为基于图的模型双向变换问题,并使用GRoundTram实现自动化的同步操作。给出了该模型同步方法的具体操作步骤,并通过一个"网上书城"的设计模型实例展示其有效性。

关键词 软件产品线,模型同步,GroundTram
中图法分类号 TP311 **文献标识码** A

Design Model Synchronization Method for Software Product Line Based on GRoundTram

QIU Lian-qi SHEN Li-wei ZHAO Wen-yun
(School of Computer Science, Fudan University, Shanghai 201203, China)

Abstract The important products of the software product line domain-level and product-level development periods are domain design model and application design model, respectively. An automatic or semi-automatic method is needed to keep synchronization between these two models during the life cycle of a software product line. Such a GroundTram-based method called SPLSync-GRoundTram was proposed in this paper. To keep synchronization between domain and application model, this method firstly converts this problem to a bidirectional transformation between two graphs, which is then solved gracefully by GroundTram. The detailed process of this method was illustrated in this paper, and an on-line-shopping case model was used to show its effectiveness.

Keywords Software product line, Model synchronization, GroundTram

1 引言

软件产品线[1]的开发过程一般可分为两个阶段:领域工程阶段和应用系统工程阶段。前者的主要任务是获取产品线核心资产,而后者主要基于核心资产定制、生成符合特定需求的制品。

在软件产品线中,设计模型是核心资产中的一种重要制品。通常情况下,根据不同的用户需求,可从同一个领域设计模型中定制得到多个应用系统设计模型。在定制之后,领域设计模型与应用系统设计模型往往会进行独立的演化。例如,应用系统设计模型在后续的软件开发过程中可能会根据用户的需求进行相应的改动,而改动过的模型可能与原始的设计模型出现不一致的情况。举例而言,为了满足某个特定的用户需求,产品工程师可能会直接向应用系统设计模型中添加一个功能模块,而领域设计模型中并不包含该模块且没有进行相应的更新,此时就出现了领域模型和应用系统模型之间的不一致情况。因此,为了保证模型之间的一致性,针对某一模型进行的变更需要被同步更新至其他模型。否则,随着产品线模型的不断演化,领域设计模型与应用系统设计模型之间的差别会愈发增大,由此导致针对各个模型的单独管理,从而增加了产品线的维护成本,致使产品线走向失败。

针对软件产品线中的领域设计模型与应用系统设计模型之间的同步问题,本文提出一种基于GroundTram的设计模型同步方法,称为SPLSync-GRoundTram。该方法旨在自动化地将应用系统设计模型的变更同步反馈至领域设计模型。SPLSync-GRoundTram的主要实现步骤包括:首先将领域设计模型翻译为图代数语言表示的描述方式,并保持二者间的一致性;其次通过GroundTram提供的双向变换功能实现在图代数语言层次上的模型描述的更新反馈;最后将图代数语言表示的更新翻译成针对设计模型的更新。基于该方法,针对"网上书城"的设计模型同步问题进行了实验,从而展示了该方法的有效性。

本文第2节对文章所涉及的基本背景及技术进行简要介绍;第3节将详细描述SPLSync-GRoundTram的实现阶段与步骤;第4节展示基于该方法的"网上书城"同步案例;第5节列举与软件产品线模型同步相关的工作;最后是本文的总结。

到稿日期:2012-05-20 返修日期:2012-08-20 本文受软件产品线开发过程改造咨询与技术支持资助。
邱连启(1986—),男,硕士生,主要研究方向为软件产品线,E-mail:qlq3763@126.com;**沈立炜**(1982—),男,博士,讲师,主要研究方向为软件工程;**赵文耘**(1964—),男,教授,博士生导师,主要研究方向为软件工程。

2 背景介绍

2.1 扩展可变性的软件产品线设计模型

基于 UML 类图的软件产品线设计模型是本文的研究对象。为了支持对领域可变性的表示，使用 UML 标准中的泛型对[2]类图进行扩展。产品线的可变性通常可分为 3 种类型，分别是可选(opt)、多选一(alt)和多选多(or)，它们的展现方式如图 1 所示。其中，可选和多选多类型的元素是通过关联关系与其父元素关联起来的，而多选一变体是通过继承关系与其父元素关联起来的。例如，图中货到付款表示一个可选类型的元素，其泛型为《opt》；查询处理表示一个多选一类型的元素，它的两个变体分别为按书名查询和按作者查询，其泛型为《alt》；特色服务为一个多选多类型的元素，即可以从排行、个性化推荐和描述元素中选择一个或者多个绑定到定制体系结构中，其泛型为《or》。

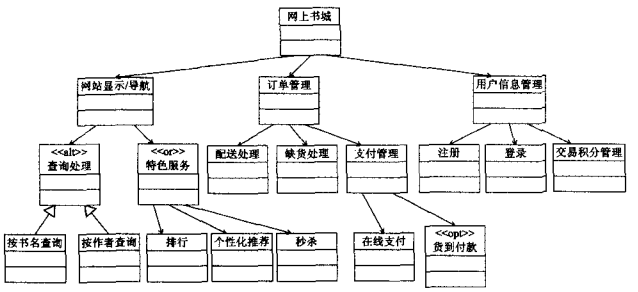

图 1 扩展泛型描述的产品线设计模型实例

在应用系统工程阶段，领域的设计模型被加以定制。此时，对于需要绑定的可选元素将其泛型《opt》清空即可，对于不需要绑定的可选元素可将其直接删除；对于需要绑定的多选一变体元素将其他变体元素及其父元素删除并用选定的变体代替其父元素的位置；对于多选多类型的元素，首先将泛型《or》清空，随后删除不需要的变体。图 2 展示了针对图 1 的领域设计模型进行定制的结果，其中被绑定的元素以红色字体标记。例如，具有多选一可变性的元素“查询处理”绑定变体“按书名查询”，具有多选多可变性的元素“特色服务”绑定了“排行”和“个性化推荐”这两个变体。另外，可选元素“货到付款”被涵盖在应用系统中。

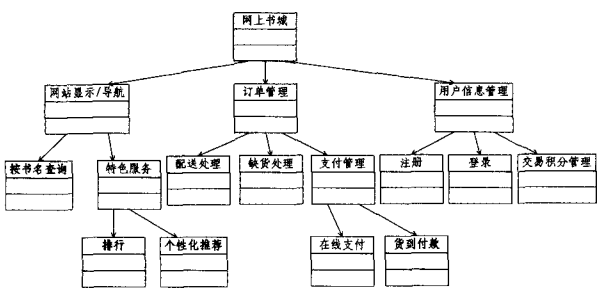

图 2 定制后的应用系统设计模型

2.2 GRoundTram

GRoundTram 是一个基于图的且支持模型双向变换的工具[3]。在定义源图到目标图的变换规则后，该工具能够实现这两个图之间的双向变换，即其中一个图的变更能够同步传播至另一个图。具体而言，GRoundTram 将源图的信息用 UnCAL[3]语言进行描述，然后使用 UnQL+[3]语句从图中根据设定的条件选取所需的信息来生成目标描述，这一过程称为正向变换。如果对目标图的信息即目标描述进行了更改，通过 GRoundTram 的反向变换就可以将更改的信息反馈到源描述或源图上。

Unql+是一个类似于 SQL、用来描述图变换的高级编程语言，是对图查询语言 UnQL[4]进行扩展而得到的。为了更方便地描述图的变换操作，UnQL+在 UnQL 的基础上实现了对图的 3 个编辑操作。因此，UnQL+主要支持两种类型的操作，第一类是 UnQL 原有的对图的查询操作，即 select-where；第二类是对图的编辑操作，即 replace-where、delete-where 和 extend-where。基于这些基本操作，UnQL+可以完成对图的基本编辑工作。

另外，UnCAL 是一个描述图代数运算的语言，并且是图查询语言 UnQL 和 UnQL+的核心部分。在实际应用中，UnQL 和 UnQL+语言均首先被翻译成由 UnCAL 语言描述的代数操作，随后被加以执行。因此，UnCAL 语言既可用来实现图的创建工作，也可用来实现图的变换和编辑工作。差异在于其语法没有 UnQL 或 UnQL+语言简单，且语义不易被一般用户理解，使用起来不够方便。例如，为了执行图的变换操作，使用 UnQL+语言时只需写出 select-where 语句来给出所要进行的变换条件即可，而使用 UnCAL 时则需要编写一个递归来遍历图中所有信息，以便在遍历的过程中对图中对每个信息进行条件判断。

在软件产品线中，应用产品的定制过程可以看作从领域模型中选择所需功能的过程。若将领域设计模型和应用系统设计模型均采用图代数语言 UnCAL 进行描述，定制过程可视为类似于 GroundTram 工具所支持的图的正向变换过程，而将应用系统模型中的更新同步反馈到领域模型的过程也就可被看作图的反向变换过程。

3 SPLSync-GRoundTram

3.1 方法概述

SPLSync-GRoundTram 的方法概览如图 3 所示。在界面部分，用户主要关注领域设计模型与应用系统设计模型及它们的变化情况。首先，应用系统设计模型是通过对领域设计模型中的可变性进行定制而得到的。在产品线生命周期中，若应用系统模型发生了变更，则通过 SPLSync-GRoundTram 的同步操作可自动得到变更后的领域模型。

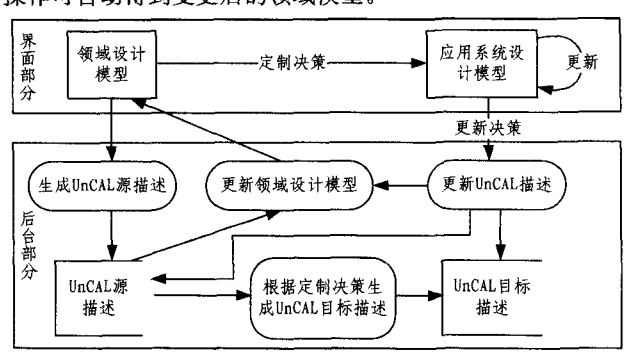

图 3 SPLSync-GRoundTram 概览

该同步过程在图 3 的后台部分中展示。SPLSync-GRoundTram 主要分为 3 个阶段，首先是模型信息收集阶段，该阶段的任务是将领域设计模型转换为一个使用 UnCAL 描述的源描述，并根据定制决策从 UnCAL 源描述生成一个基于 UnCAL 描述的目标描述，即应用系统设计模型的 UnCAL

表示；其次是获取更新策略阶段，该阶段的主要任务是获取用户对应用系统设计模型的变更，即用户执行了哪些更新操作；最后是更新反馈阶段，根据阶段一收集到的模型信息以及阶段二获取到的更新策略，该阶段首先完成对 UnCAL 源描述和 UnCAL 目标描述的更新，随后将对 UnCAL 源描述的更新转换为对领域设计模型的更新。通过这 3 个阶段，可实现一个由应用系统模型变更所驱动的模型同步的过程。

3.2 阶段 1：收集模型信息

该阶段的任务可分为两个子活动：第一，将领域设计模型翻译成 UnCAL 源描述；第二，根据定制决策，从 UnCAL 源描述生成 UnCAL 目标描述。

将领域设计模型翻译成 UnCAL 源描述较为简单，只需从设计模型即类图中读取相应的信息，然后按照预定义的格式生成对应的 UnCAL 描述元素。描述 UnCAL 模式的语言被称为 KM3[3]，本文使用的模式如表 1 所列。每个类对象 Clazz 有 3 个属性：名字 name、可变性类型 varType 以及子元素数组 children。表 2 则给出了符合该模式，且针对图 1 所示的领域设计模型所生成的 UnCAL 描述片段。该片段包含了两个元素的 UnCAL 描述：多选一元素“查询处理”和可选元素“货到付款”。由于篇幅原因，本文并未将所有元素的 UnCAL 描述在此列出。

表 1 UnCAL 类图模式

```
package UpdateBack {
    datatype String;
    class Calzz {
        reference name:String;
        reference varType:String;
        reference children [0-*]:Clazz;
        }
}
```

表 2 UnCAL 片段

```
&clazzSearch:=
{ name:"查询处理",
  varType:"alt",
  children:&clazzSchName,
  children:&clazzSchAuthor
}
&clazzPayOnSite:=
{ name:"货到付款",
  varType:"opt"
}
```

在第 2 个子阶段中，根据定制决策，从 UnCAL 源描述生成 UnCAL 目标描述。首先，需要根据定制决策生成与之相对应的 UnQL+语句，然后通过 GroundGram 工具执行生成的 UnQL+语句来得到定制后的 UnCAL 目标描述。为了生成 UnQL+语句，将定制决策分成 3 种情况分别进行处理：

(1)可选元素是否进行了绑定，如果没有进行绑定，则生成的 UnQL+语句中不选择该元素即可，否则在生成的 UnQL+语句中将元素的 varType 字段清空。

(2)多选一类型，在生成的 UnQL+语句中将 varType 为 alt 的父元素换为绑定的子变体元素。

(3)多选多类型，生成的 UnQL+语句中将父元素的 varType 字段清空，且过滤没有选中的子元素。

3.3 阶段 2：获取更新策略

常见的对模型更新进行反馈的场景是：用户每次只更新模型中的一小部分，当积累了一组小粒度更新之后，用户可一次性触发同步操作，而不是每次当模型有了更新才进行反馈。在同步过程中，该方法的输入是最新的应用系统设计模型以及原始的领域设计模型。此时，需要根据最新的应用系统设计模型和未更新的且经过保存的原始 UnCAL 源描述来获取更新策略。为此，本文提出表 3 所列的获取算法来实现该功能。算法中考虑两种更新策略，即元素的添加和删除。更新策略的内容包括：类型(添加或删除)、添加或被删除元素的名字(本文使用元素的名字作为元素的唯一标记)、添加或被删除元素的父元素的名字。获取添加类型策略的流程如表 3 所列，遍历最新的 UnCAL 目标描述中的每个元素，如果该更新前的 UnCAL 目标描述中不包含该元素，则添加一个更新策略。由于篇幅所限，这里只列出了获取添加策略的流程，获取删除策略的流程与获取添加策略的流程基本相同，只是将循环中的 UML model 变成了 UnCAL model，且策略的类型变成了 delete。

表 3 添加策略获取算法

```
算法：更新策略的获取
输入：最新的应用系统设计模型与更新前的 UnCAL 目标描述
输出：更新策略
过程：
for(every element in UML model)
  do
    get element name;
      if(name is not in UnCAL model)
        do
          get parent element name
        add new strategy(add,name,parent name)
      end if
end for
```

随后，对上一步所得到的更新策略做进一步的处理。首先，对所得到的删除策略按照以下方式进行排序：如果存在一个策略 A1 的 *parent name* 等于另一个策略 A2 的 *name*，则将 A1 排在 A2 的前面，这样就可以使得在进行删除操作时总是从底层的元素开始，然后向上层进行删除。其次，对所得到的添加策略也进行排序，只是排序的方式与对删除策略的排序方式相反，即如果存在一个策略 A1 的 parent name 等于另一个策略 A2 的 name，则将 A2 排在 A1 的前面，这样就使得添加操作总是从上层开始，然后向下层进行添加操作。最后对每个添加策略进行如下处理：

(1)在 UnCAL 源描述中获取父元素的父元素(如果存在)，即祖父元素。

(2)获取由(1)得到的祖父元素的所有多选一类型的子元素(如果存在)。

(3)给出一个列表，让用户明确添加的元素是哪个元素的子元素，该列表的内容是以下元素的名字：策略中父元素的名字和由(2)得到的元素的名字。

(4)根据(3)中用户的选择更改当前所处理的策略中的 parent name 字段。

3.4 阶段 3：更新反馈

该阶段的任务也可分为两个子活动：UnCAL 描述的更新以及将对 UnCAL 源描述的更新转换成针对领域设计模型的更新。

首先，UnCAL 描述的更新包括针对 UnCAL 源描述和 UnCAL 目标描述的更新。主要任务为根据上一阶段得到的更新策略生成并执行相应的 UnCAL 语句，从而更新 UnCAL

源描述和 UnCAL 目标描述。

对每一个删除策略进行如下处理：首先生成一个根据元素名字进行删除的 UnCAL 语句，该语句只针对 UnCAL 目标描述执行；然后根据被删除元素在 UnCAL 源描述参考体系结构中的可变性类型分两种情况进行处理：如果被删除元素是一个必选类型的元素，则生成一个更新 varType 字段的 UnCAL 语句，该语句将被删除元素的 varType 字段设置为 opt；否则，不需要执行任何操作，该语句只针对 UnCAL 源描述执行。

对每一个添加策略进行如下处理：(1)根据添加元素的名字生成一个 UnCAL 表示的需要添加的子元素。(2)在 UnCAL 目标描述中根据父元素的名字找到要添加子图的位置。(3)在找到的位置通过 GroundTram 同时向 UnCAL 目标描述和 UnCAL 源描述添加新的子元素。(4)如果被添加元素的父元素在 UnCAL 源描述中的类型为可选或者必选，则生成对 UnCAL 源描述更新 varType 字段的 UnCAL 语句，该语句将被添加元素的 varType 类型设置为 opt。

其次，将 UnCAL 源描述的更新翻译成针对领域设计模型的更新。由于前面的更新反馈操作并不会删除领域设计模型中的任何元素，因此不应该出现某元素存在于领域设计模型而不存在于 UnCAL 源描述中的情况。对每个元素来说，如果该元素同时出现在设计模型和 UnCAL 描述中，且在两类模型中该元素的可变性类型相同，则此时该元素在两个模型中的信息是一致的，不需要采取任何动作。因此，仅存在如下两种情况需要被处理：(1)该元素同时出现在 UnCAL 源描述和领域设计模型中，且在两种模型中的可变性类型不同。此时，将设计模型中该元素的可变性类型设置成 UnCAL 描述中的可变性类型即可。(2)该元素只在 UnCAL 描述中出现，则在设计模型中相应的位置添加该元素。

4 实例展示

为了验证 SPLSync-GRoundTram 方法，将图 1 中的领域设计模型作为输入，并且使用 2.1 节提到的定制决策对图 1 中的领域模型进行定制，得到的应用系统设计模型如图 2 所示。在此基础上，我们将详细介绍基于 SPLSync-GRoundTram 的模型同步过程。

第 1 步　使用方法阶段 1 中提及的技术得到 UnCAL 源描述和 UnCAL 目标描述。该部分比较简单，主要是将领域设计模型中的元素翻译成 UnCAL 表示的元素的过程和将定制决策翻译成 UnQL+ 的过程。所得的 UnCAL 源描述可参见表 2，得到的 UnCAL 目标描述形式与之类似。

第 2 步　获取更新策略。假设应用系统设计模型发生了如下更新：元素“按书名查询”被删除，同时在该位置添加了元素“按书名查询”；元素“排行”被删除，同时在该位置添加了元素“心愿单”；元素“货到付款”被删除，同时在该位置添加了元素“pos 机刷卡”；元素“交易积分管理”被删除。更新后的应用系统设计模型如图 4 所示。通过表 3 给出的算法，可以得出初步的更新策略，结果如表 4 所列。由于添加或删除更新策略间不存在直接的层次关系，因此接下来的排序操作不对得到的更新策略产生影响。而在此后给出的选项列表中，假设用户选择了元素“查询处理”作为元素“按出版社查询”的父节元素，此时则需要对表 4 中的更新策略进行相应的修改，即(“按出版社查询”，“网站显示/导航”)更新为(“按出版社查询”，“查询处理”)。

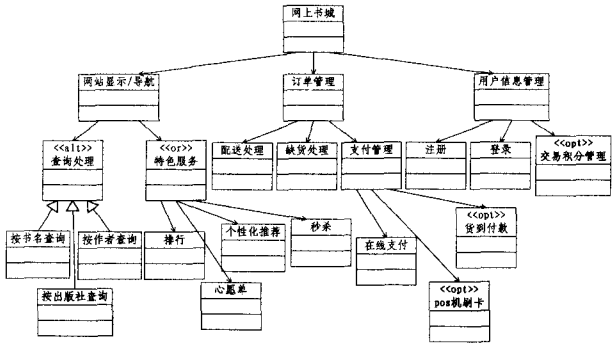

图 4　变更后的应用系统设计模型

表 4　更新策略

省略了 type 字段，格式为(name，parent name)
add：(按出版社查询，网站显示/导航)，(心愿单，特色服务)，(pos 机刷卡)。
delete：(按书名查询，网站显示/导航)，(排行，特色服务)，(货到付款，支付管理)，(交易积分管理，用户信息管理)。

第 3 步　更新反馈。首先，将第 2 步得到的更新策略翻译成一系列对应的 UnCAL 语句，然后将得到的 UnCAL 语句使用 GroundTram 工具对 UnCAL 源描述和目标描述进行更新。更新后的 UnCAL 源描述的部分结果如表 5 所列。其中，为多选一元素“查询处理”增加了一个子变体元素“按出版社查询”(clazzSchPublisher)，添加了一个新的可选元素“pos 机刷卡”，同时元素“交易积分管理”的类型变为可选。最后，使用更新翻译算法将 UnCAL 描述的更新翻译成针对设计模型的更新，变更后的领域设计模型如图 5 所示。可以看到，新增元素“按出版社查询”、“心愿单”、“pos 机刷卡”在正确的位置以正确的类型被添加到了领域设计模型中，而被删除的元素“交易积分管理”的可变性类型也被更改为了可选类型。

表 5　更新后 UnCAL 源描述（部分）

```
&clazzSearch:=
  { name:"查询处理",
    varType:"alt",
    children:&clazzSchName,
    children:&clazzSchAuthor,
    children:&clazzSchPublisher
  },
&clazzPos:=
  { name:"pos 机刷卡",
    varType:"opt"
  },
&clazzCredit:=
  { name:"交易积分管理",
    varType:"opt"
  }
```

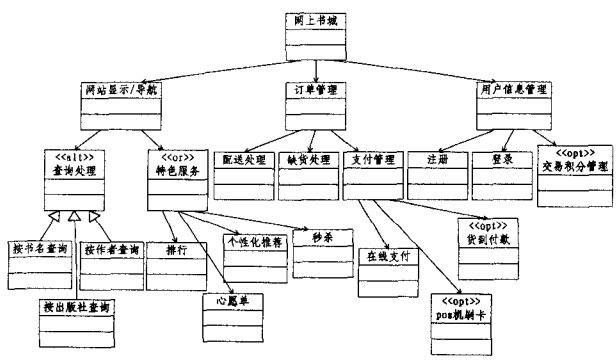

图 5　更新后的领域设计模型

(下转第 231 页)

结束语 本文以 hadoop 平台上的并行广度算法(parallel breath-first search)为基础,提出了一种基于 HBase 的并行 BFS 方法。该方法将 HBase 数据库的表数据存储功能和 HBase 的 Coprocessor 的处理功能加入到并行 BFS 求取单源最短路径树的过程中,使得新的 BFS 方法只用 map 阶段,有效地提高了最短路径树提取的运算效率。最后利用 Watts-Strogatz 模型对网络数据建模,通过对数据运行结果的分析,证实了改进后算法的正确性和高效性。

参 考 文 献

[1] Lakshman A, Malik P. Cassandra : A Decentralized Structured Storage System[C]//SIGOPS. 2010

[2] HBase Homepage[OL]. http://hbase. apache. org

[3] Chang F, et al. Bigtable: A Distributed Storage System for Structured Data[C]//OSDI. 2006

[4] 许春聪,黄小猛. 分布式文件系统存储介质评测与分析[J]. 软件学报,2010,33(10):1873-1880

[5] Lin J, Dyer C. Data-Intensive Text Processing with MapReduce, ser. Synthesis Lectures on Human Language Technologies[M]. Morgan & Claypool Publishers, 2010

[6] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters[J]. Commun. ACM, 2008, 51(1): 107-113

[7] 周国亮,陈红. 基于图形处理器的并行方体计算[J]. 软件学报,2010,33(10):1788-1798

[8] Dean J. Large-scale distributed systems at google: Current systems and future directions[C]//The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware. 2009

[9] Dijkstra E W. A note on two problems in connexion with graphs [C]//Numerische Mathematik. 1959: 269-271

[10] Cormen T H, Leiserson C E, Rivest R L. Introduction to Algorithms[M]. Cambridge, Massachusetts: MIT Press, 1990

[11] Kleinberg J. The small-world phenomenon: An algorithmic perspective[C] // Proceedings of the 32nd ACM Symposium on Theory of Computing. 2000: 163-170

(上接第 218 页)

5 相关工作

目前,关于软件产品线中模型同步方法的研究多是基于软件配置管理(SCM)[5]的概念,即如何管理复杂软件系统中不同版本之间的同步问题。在传统的软件配置管理工作中主要关注的是单个产品的同步管理,且版本管理工具往往在其中起着重要的作用,有些方法在传统的软件配置管理过程中被证明了是可用的。因此,有些研究主要是关注如何将原来用于管理单个产品同步的方法进行扩展以管理产品线的同步[6,7],而有些研究则将版本管理工具作为软件产品线同步管理的基础[6,8]。同时,关于软件产品线同步管理的研究多为使用"小受众"的模型来表示领域设计模型,如 xADL[8-10]。最后,关于如何使用 UML 模型提供的元素来表示软件产品线中的可变性概念,可以参考文献[2]。

结束语 为解决软件产品线中以 UML 类图为表示方法的领域设计模型与应用系统设计模型之间的同步问题,本文提出一种基于双向变换工具 GRoundTram 的设计模型同步方法 SPLSync-GRoundTram,为模型同步提供了一种新的思路。该方法将领域和应用系统设计模型均翻译成图代数语言,并在更加形式化的层次上实现自动化的模型同步。依照 SPLSync-GRoundTram 的方法概览,模型同步由 3 个阶段的活动实现。基于该方法,我们针对一个"网上书城"的设计模型进行了同步操作,以验证方法的有效性。

参 考 文 献

[1] Kang K, Cohen S, Hess J, et al. Feature-Oriented Domain Analysis(FODA) Feasibility Study [R]. Technical Report CMU/SEI-90-TR-021. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1990

[2] Gomaa H. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures [M]. Boston: Addison Wesley, 2005

[3] GRoundTram Version 0. 9. 2 User Manual [OL]. http://www.biglab.org/download.html, 2012-05-20

[4] Buneman P, Fernandez M F, Suciu D. UnQL: a query language and algebra for semistructured data based on structural recursion. VLDB [J]. The VLDB Journal, 2000, 9(1): 76-110

[5] Westfechtel B, Conradi R. Software Architecture and Software Configuration Management[C]//10th International Workshop on Software Configuration Management: New Practices, New Challenges, and New Boundaries (SCM 10). Toronto, Canada, 2001: 19-26

[6] Gurp J, Prehofer C. Version Management Tools as a Basis for Integrating Product Derivation and Software Product Families [C]//Proceedings of the Workshop on Variability Management - Working with Variability Mechanisms at SPLC. 2006: 48-58

[7] Yu L, Ramaswamy S. A Configuration Management Model for Software Product Line [J]. INFOCOMP Journal of Computer Science, 2006, 5(4): 1-8

[8] Peng Xin, Shen Li-wei, Zhao Wen-yun. An Architecture-based Evolution Management Method for Software Product Line [C]// Software Engineering and Knowledge Engineering. 2009, 09: 135-140

[9] Chen P, Critchlow M, Garg A, et al. Differencing and Merging within an Evolving Product Line Architecture[C]//PFE. 2003, 03: 269-281

[10] Garg A, Critchlow M, Chen P, et al. An Environment for Managing Evolving Product Line Architectures [C]//ICSM. 2003, 03: 358-367