一种基于类间关系动态分析的领域框架和构件提取方法

彭 鑫, 赵文耘

(复旦大学 计算机科学与工程系 软件工程实验室, 上海 200433)
E-mail: pengxin@fudan.edu.cn wyzhao@fudan.edu.cn

摘 要: 框架和构件提取是面向复用的软件再工程的一个重要目标. 在分析遗产系统中框架和构件的存在方式的基础上, 提出了一种领域框架和构件的提取方法. 该方法首先标识框架骨干类, 然后以这些类为原点向外进行扩展得到整个框架实例, 并通过抽象和精化得到领域框架. 从框架实例分离的那些类上继续进行扩展可以获取遗产构件. 整个扩展过程都是以所考察的类与当前类簇的整体关系作为判断依据, 因此划分更加合理.

关 键 词: 软件复用; 构件; 软件框架; 再工程; 构件提取; 度量
中图分类号: TP31 **文献标识码**: A **文 章 编 号**: 1000-1220(2007)11-1928-04

Domain Framework and Component Extraction Method Based on Dynamic Analysis of Relations Between Classes

PENG Xin, ZHAO Wen-yun
(*Computer Science and Engineering Department, Fudan University, Shanghai* 200433, *China*)

Abstract: Extraction of framework and components is an important goal of reuse-oriented software reengineering. In this paper, an extraction method is proposed on analysis of existing mode of domain framework and components in legacy systems. In the method, skeleton classes of the framework are firstly identified, and then spread is performed from these classes to get the whole framework instance. Finally, the domain framework can be abstracted from the instance. Spread can continue from those classes, where the framework instance separates, to extract legacy components. All of the spread processes are based on the dynamic analysis of the overall relationship between the examined class and the class cluster, so the partition is more reasonable.

Key words: software reuse; component; software framework; reengineering; component extraction; software measurement

1 引 言

面向复用的软件再工程以构件化为目标, 着眼于从遗产系统中进行框架和构件提取. 获得的实现框架经过抽象和精化后可以作为应用产品族框架. 而获取的构件一部分具有领域内的复用价值, 可以作为领域公共构件; 另一部分具有特定应用特征, 需要在此基础上进行抽象得到构件模板以指导新的应用系统构件开发. 在此基础上一方面可以对遗产系统进行重构以提高软件质量、降低维护成本, 另一方面也可以在应用族框架上复用公共构件, 定制特定应用构件, 从而快速构造出新的应用系统.

复用业务定义了三种软件工程过程: 应用族工程、构件系统工程和应用系统工程[1]. 相应的应用系统中包含三种类型的构件: 框架构件、领域构件和特定应用构件. 这三类构件在整个应用族中扮演着不同的角色. 构件系统通过门面(facade)导出服务[1], 应用族工程将这些门面组织成领域构架, 为构件系统工程和应用系统工程的交互提供"装配点". 框架是实例化的DSSA(特定于领域的构架), 是实现大粒度软件复用的有效途径. 同时, 框架通过扩展点组装用户开发的构件, 以处理领域变化性[2].

综合文献[1]和[2]中的观点, 我们认为应用系统中存在的框架由体现领域高层结构共性的框架构件构成. 框架构件中包括一部分框架内部实现构件, 它们构成了框架的核心内容; 另外一部分构件是抽象构件或者门面构件, 它们是应用构件的组装点, 为领域的变化性提供接口. 对于遗产系统, 其中有一部分框架构件可能已经被实例化为应用构件. 这种情况在许多软件系统中存在, 主要原因是系统设计时只考虑了特定应用的需要, 因此造成应用族框架与应用构件间的界限不清晰. 考虑到框架复用的需要, 必须对这种框架进行精化, 将其中与特定应用相关的成分分离出来, 取而代之以体现领域共性的构件门面或抽象构件.

文献[3]认为构架恢复有两种目标, 即获取曾经指导了遗产系统开发但已经遗失的软件构架以及获取更加符合当前系统的软件构架并推动以后系统的演化. 目前在构架恢复(Architecture Recovery)以及构件获取上已经有一些比较系统的方法. 文献[3]通过一系列迭代开展的过程进行构件恢复, 其

收稿日期: 2006-07-26 基金项目: 国家"八六三"计划基金(2005AA113120)资助; 国家自然科学基金(60473061)资助; 上海市科委科技攻关项目(04DZ15022). 作者简介: 彭 鑫, 男, 1979 年生, 博士、讲师, 研究方向为构件技术、软件体系结构、软件再工程; 赵文耘, 男, 1964 年生, 教授、博士生导师, CCF 高级会员, 研究方向为软件工程及电子商务、企业应用集成(EAI).

中的核心思想是通过识别构架关注点、层次关系、构件簇、连接器、接口等一系列步骤获取系统构架. 文献[4]从高层和低层重构两个方面阐述了面向框架的面向对象软件再工程方法. 文献[5]通过构件挖掘和构件质量度量从遗产系统中获取构件. 文献[6]以一种进程的观点研究体系结构恢复, 提出从动态运行时的进程 ID 与静态代码片断之间的映射关系出发, 增量地构造进程结构图并通过一种切片算法获取进程模块内部的类结构, 从而为目标系统提供一种基于进程的高层结构视图. 文献[7]将体系结构恢复建模为图匹配问题, 通过迭代地执行用户定义的图模式(体现了一种体系结构约束)与体现系统实现信息的源代码实体关系图之间的匹配来实现体系结构的恢复. 文献[8]将特征模型用于程序理解和体系结构恢复, 通过特征模型对功能性需求进行建模, 在此基础上通过探测源代码与需求之间的关系来支持体系结构的恢复.

以上这些架构恢复工作都只面向特定应用, 以辅助程序理解或构件提取为目标. 一些较新的工作(如文献[8])将需求模型引入到体系结构恢复中并将重构纳入体系结构恢复的目标, 但仍然没有系统地考虑面向应用族的框架提取以及重构的需要. 然而在现实的领域开发中, 完全正向的复用式开发并不多见, 而基于已有的多个独立领域应用总结提炼领域框架和可复用构件并用于新的应用开发则是普遍现象. 因此面向应用族自动或半自动的逆向工程方法支持对于现实中的特定领域软件开发具有十分重要的意义. 针对这一状况, 本文在分析遗产系统中框架和构件的存在方式基础上, 提出了一种面向领域的框架和可复用构件的提取方法, 从而为面向应用族的软件复用提供逆向工程支持. 该方法首先标识框架骨干类, 然后以这些类为原点向外进行扩展得到整个框架实例, 并通过抽象和精化得到领域框架. 从框架实例分离的那些类上继续进行扩展可以获取遗产构件. 整个扩展过程以所考察的类与当前类簇的整体关系作为判断依据, 因此划分更合理.

2　框架及构件提取方法

文献[4]指出底层的重构在某种程度上可以机械地进行, 而高层重构则必须在人工干预下进行. 我们的框架和构件提取过程着眼于人工干预和自动处理机制的良好结合, 以框架的分离及精化、构件的识别和可变性分析为主线, 在此基础上对遗产系统进行重构并为应用族中新系统的开发打下良好的基础. 这个方法主要包括以下几个步骤:

(1) 标识遗产系统框架中的骨干类, 并以它们为原点向外扩展得到完整的框架实例;

(2) 对框架实例进行动态分析, 识别框架中的抽象构件和构件系统门面以进行精化;

(3) 从框架分离的那些"点"上继续分析, 获取外部应用构件和公用构件并进行可变性分析;

(4) 按照应用族框架, 对逆向过程中所发现的与分层构架不符的地方进行重构.

这些步骤中, 第1 和第3 步中的扩展分析都需要基于类间关系分析的扩展算法支持, 具体算法将在第3 节介绍.

2.1　标识框架骨干

框架骨干是指那些最能体现整个系统脉络的类, 这些类构成了框架的核心. 以下两种方法可以有效的帮助我们标识构件骨干:

2.1.1　借助高层模型　业务模型中的高层用例对应于应用系统或构件系统中的一个或多个用例[1]. 虽然遗产系统可能是在完全没有应用族框架概念的情况下开发出来的, 但其中必然包含体现高层结构特征的元素. 在领域专家的帮助下, 我们完全有可能找到与高层用例(例如用例图或者活动图)对应的那些框架片断.

2.1.2　基于业务实体的层次分析　层次体系结构将系统的不同特征组织在不同层次上[9]. 因此不同层次的关注点各不相同. 代表业务实体的实体构件一般处于层次结构的下层, 而体现应用接口的框架构件一般处于上层, 这一点在企业信息系统中尤为明显. 根据这一特点, 我们可以从能与业务实体对应起来的实体类开始, 按照类间关系自底向上根据构件依赖关系标识层次. 这个过程中处于较高层次上的那些类就有可能是框架成分.

通过这两种方法获得的只是实例框架片断, 必须经过扩展和精化才能获得应用族框架.

2.2　框架扩展

框架是一个完整的结构, 因此需要对获得的框架骨干进行扩展, 即: 从骨干类出发, 沿着类间关系图进行动态分析, 不断地将周边框架类纳入, 从而获取完整的框架实例. 框架扩展有两个尺度, 一个是关联尺度, 一个是连通尺度. 关联尺度是指按照某个扩展阈值R 考察相关类的加入, 从而体现框架构件的内聚. 连通尺度是指扩展过程着眼于将片断串联成完整的框架, 避免框架的不完整. 关联尺度的作用在于以骨干类为原点标识框架构件, 而连通尺度可以帮助我们发现遗漏的框架片断. 基于类间关系的动态扩展是其中的关键, 将在第3 部分介绍. 当扩展过程结束后, 我们就得到了完整的实例框架, 其中包含了一些应用实例内容, 需要进一步进行精化以恢复应用族框架.

2.3　框架实例的精化

框架实例包含特定应用信息, 还需要进行精化以获得领域框架. 文献[2]中将角色定义为"定义了接口以及调用规约而没有完全实现的构件", 认为角色反映一个应用构件在协作中所承担的责任, 起到"占位符"的作用. 这里的角色对应于框架构件中的门面和抽象构件. 框架精化要做的工作就是标识实例框架中的应用相关成分, 识别其中的"占位符", 从而抽取框架结构.

实例框架中隐含的应用相关成分很难通过静态分析获得, 需要借助动态的系统分析. 动态分析必须针对应用族中不同应用系统或同一应用系统中结构相似的不同子系统进行. 通过动态分析可以发现框架实例中特定应用相关成分, 它们一般只在特定应用中涉及. 在此基础上, 我们可以标识它们在框架中与前后其它部分的接口和关联方式. 如果特定应用成分通过类似的高层类体现外部视图则仅保留其高层类作为门面; 如果是通过类似的操作或消息传递体现对外视图, 则对接口进行抽象得到抽象构件, 使之取代框架中的实例成分. 通过

这样的精化处理, 我们将得到由内部实现构件、门面构件和抽象构件三种构件组成的应用族框架

2.4 构件获取

框架构件扩展停止的边界点上体现了其它构件的装配关系. 从这些地方继续向系统外围扩展就可以进行构件分解. 每一次扩展停止就意味着一个候选构件的产生以及新的扩展的起点.

当系统中所有的类都划入某个类簇后我们就得到全部类簇划分了. 这些类簇就是候选构件. 它们中有的具有领域内的复用价值, 是领域共性的基础成分; 有的则体现了特定应用特征, 复用价值较小, 但可作为同一应用族中其它系统构件开发的参照.

3 基于类间关系分析的动态扩展

框架扩展以及构件获取过程都是基于类间关系分析的动态扩展过程. 扩展过程从某些起点类开始, 沿着类间关系图逐一考察周围的类是否应该链入当前类簇.

文献[5]利用类间关系的带权无向图的连通分析进行构件划分. 文献[9]通过聚类分析识别业务构件. 这两种方法的一个局限性是每次筛选都只考虑了两个类之间的关系, 而构件的内聚往往体现在类簇内的多边关系上. 基于类间关系的动态扩展方法以单个类与当前类簇的整体关系作为衡量标准, 因此得到的类簇划分结果更合理.

3.1 动态扩展算法

动态扩展算法基本步骤如下:

(1) 确定一个或多个扩展的起始类簇P. 在框架扩展过程中, 识别出的骨干类构成了起点类簇. 构件获取过程中, 框架分离点构成了若干个独立的扩展起点;

(2) 确定关联值计算方法和扩展阈值R;

(3) 计算每一个与P 有直接关联的类C 与P 之间的关联值, 当关联值大于R 时将C 加入P 中;

(4) 重复进行3 直到没有新的类加入P 为止;

(5) 本次扩展中经过考察未加入P 的那些类成为新的扩展起点, 可以继续进行扩展以获取其它类簇.

在这个扩展过程中, 每个与P 相关的类都会多次被考察到, 这样某个类C 就有可能经过多次考察后才加入P. 因为随着其它类加入P, 类C 与P 的多边关联值可能会越来越大.

3.2 关联值算法

关联值的计算方法主要应该考虑到体现类C 与类簇P 的多边关联度. 首先定义四个与关联度计算相关的因子:

$$RCP = \sum_{i\in P} RCi, RCN = \sum_{i\in N} RCi$$

$$refCP = \left|\{A \mid (A\ refers\ C) \wedge ((A\ refers\ P) \vee A \in P)\}\right|$$

$$refC = \left|\{A \mid A\ refers\ C\}\right|$$

其中N 为全体类的集合, RCi 表示类与 i 类C 的原子关联度(原子关联度的计算方法在文献[5]和[9]中都有论述, 可以根据实际情况选择), $refers$ 操作符表示有向的关联关系. 这里只考虑有向关联, 即考察类C 是否应该扩展入类簇P 时只考虑类C 被其它类关联的情况. 四个关联度计算因子定义中: RCP 和RCN 分别表示类C 与类簇P 和全体类N 中各类关联值的总和; $refCP$ 表示同时关联类C 和类簇P 的类的数量; $refC$ 表示关联C 的类的数量. 在此基础上, 我们可以定义三种关联度以及总的关联度计算公式如下:

$$1.\ \frac{RCP}{RCN} \qquad 2.\ \frac{refCP}{refC} \qquad 3.\ \frac{|N| - refC}{|N|}$$

三种关联度指标的含义如下:

· $ref\ 1$ 表示类C 与P 直接关联的密切度, 用C 与P 中各个类关联值的总和占总关联值的比例来衡量;

· $ref\ 2$ 表示类C 与P 协作提供服务的概率, 用同时关联C 与P 的类占关联类C 的类总数的比例来衡量;

· $ref\ 3$ 表示类C 的通用程度, 这是一项负指标, 通用程度越高该指标越低.

在此基础上计算出来的类C 与P 的总关联度ref 是这三个关联度指标的加权和加权系数的取值可以根据情况选定, 一般前两个关联度指标的权重较大, 是主要决定因素. 例如我们在应用中采用0.4、0.4、0.2 的权重取得了良好的效果.

3.3 扩展点合并

当扩展产生多个扩展原点时(例如上一次扩展在多个周边类上停止), 可能导致扩展方向过于分散. 例如在图1 中, 如果上一次扩展停止在Origin1 和Origin2 这两个类上, 那么分别以这两个类为原点进行扩展就会发现很难决定Class1-3

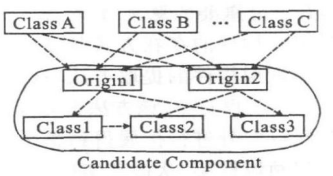

图1 扩展过程中的原点合并

Fig. 1 Merging of origins of the spread

的归属, 因为它们与这两个原点具有相似的关联程度. 不难发现, 这种情况的出现是因为Origin1 和Origin2 分别进行扩展, 而它们对外提供的服务接口总是在同一环境下使用. 这两个原点如果合并作为起始类簇进行扩展则可以得到图1 中的候选构件. 这样的扩展结果更接近于我们的需要, 因为Origin1 和Origin2 的接口在很大程度上体现为一个构件的两种服务接口. 因此有必要在3.2 中的扩展方法基础上增加扩展原点的合并分析过程. 我们认为两个扩展原点如果满足以下两个条件则应该将它们合并:

(1) 它们被一起使用的概率较高, 可以用类似于3.2 中公式2 的方法度量.

(2) 它们所关联的类集合中相同成分的比例较高, 可以通过对它们直接或间接关联的类集合的成分分析度量.

从框架构件扩展停止点开始的构件识别过程中, 产生的每一批新的扩展起点先进行合并分析, 经过整合后各自开始扩展过程. 只要选择的阈值合理(例如我们发现以100 为满值则60-90 较合适), 一般不会出现类簇之间的类成分重叠.

3.4 算法复杂度分析

扩展算法在初始阶段计算并保存各个类的关联类及原子

关联度列表用于后续关联度计算. 这也是文献[5]、[9]中的方法所必需的. 这部分的算法复杂度是O (n²).

扩展过程中的关联度计算公式(3. 2 节)中, 第三个对于类C 是定值, 另两个中只有*RCP* 和*refCP* 两个因子需要在动态扩展过程中进行计算. *RCP* 和*refCP* 的计算只需要对原子关联度列表进行 |P |次查询和一次相加计算. 这样, 考虑所有类都被划到某个类簇的计算复杂度为O (n²), 其中主要是查询操作. 由此可见, 算法初始阶段计算出关联度列表后, 后续的动态扩展过程并不会增加太大的计算量.

4　实例研究

这部分将通过一个在线宠物销售系统的局部实例来说明该方法的有效性. 图2 描述了该系统的一个局部类间关系图.

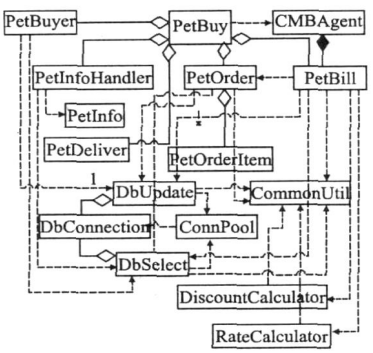

图2　一个在线宠物商店系统片断

Fig. 2　Instance fragment of an online petstore

与该系统属于相同领域(在线购物领域)的还有网上缴费、网上书店等B2C 模式的商业应用. 通过高层用例分析可知该领域核心业务包括商品浏览、订单管理、交易支付等, 因此确定PetBuy和CMBAgent作为骨干类. 进行框架扩展、精化以及

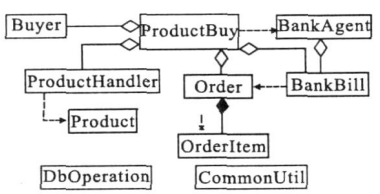

图3　框架和构件提取结果

Fig. 3　Result of framework and component extraction

构件提取后获得如图3 所示的结果. 其中, 公共构件包括DbOperation 和CommonUtil, 前者封装了与数据库操作相关的四个类. 图3 中的框架是对完整的框架实例进行精化的结果, 例如其中CMBAgent(招商银行支付代理)被抽象为代表各种银行B2C 支付代理接口的BankAgent, 宠物购买订单类PetOrder 被抽象订单类Order 代替. 另外, 原系统中的手续费、折扣计算等类被识别为特定应用构件, 并在框架中的BankBill 类上留下相关的装配点供特定应用进行装配.

5　结　论

本文提出了一种系统地从遗产系统中获取实现框架和构件的方法. 通过分解和获取过程, 我们可以获得可复用的实现框架和公共构件. 按照框架上的应用构件装配接口, 同时复用公用构件, 我们可以更容易地开发出新的应用系统. 很多情况下, 原始的遗产系统由于设计缺陷无法获得较好的框架和构件提取结果. 此时各步骤还应该包含对遗产系统的重构过程, 以使得遗产系统本身更符合清晰的层次结构.

我们对遗产系统重构的研究还不是很充分. 一些设计较差的遗产系统还无法直接运用这种方法进行分析. 今后我们将在这些方面继续开展工作.

References:

[1] Ivar Jacobson, Martin Griss, Patrik Jonsson. Software reuse: : architecture, process and organization for business success[M]. ACM Press Books, ISBN: 0-201-92476-5. First printed 1997.

[2] Liu Y, Zhang S K, Wang L F, et al. Component-based software frameworks and role extension form [J]. Journal of Software, 2003, 14(8): 1364-1370.

[3] Duenas J C, Lopes de Oliveira W, et al. Architecture recovery for software evolution[A]. Proceedings of the Second European Conference on Software Maintenance and Reengineering [C], 1998, 113-120.

[4] Serge Demeyer, Stéphane Ducasse, Robb Nebbe, et al. Using restructuring transformations to reengineer object-oriented systems[R]. May 1997, technical report. http://www. iam. unibe. ch/~ famoos/Deme97z/finalWCRE. pdf.

[5] Zhou Xin, Chen Xiang-kui, Sun Jia-su, et al. Software measurement based reusable component extraction in object-oriented system [J]. Acta Electronica Sinica, 2003, 31(5): 635-649.

[6] Li Qing-shan, Chu Hua, Hu Sheng-ming, et al. Architecture recovery and abstraction from the perspective of processes[A]. Proceedings of the 12th Working Conference on Reverse Engineering[C], 2005, 57-66.

[7] Kamran Sartipi, Kostas Kontogiannis. On modeling software architecture recovery as graph matching[A]. Proceedings of the International Conference on Software Maintenance[C], 2003, 224-234.

[8] Ilian Pashov, Matthias Riebisch. Using feature modeling for program comprehension and software architecture recovery[A]. Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems [C], 2004, 406-418.

[9] Xu W, Yin B L, Li Z Y. Research on the business component design of enterprise information system [J]. Journal of Software, 2003, 14(7): 1213-1220.

附中文参考文献:

[2] 刘　瑜, 张世琨, 王立福, 等. 基于构件的软件框架与角色扩展形态研究[J]. 软件学报, 2003, 14(8): 1364-1370.

[5] 周　欣, 陈向葵, 孙家骕, 等. 面向对象系统中基于度量的可复用构件获取机制[J]. 电子学报, 2003, 31(5): 649-635.

[9] 徐　玮, 尹宝林, 李昭原. 企业信息系统业务构件设计研究[J]. 软件学报, 2003, 14(7): 1213-1220.