以项目为中心的面向对象复用支持*

陈小群 邵维忠 梅 宏 杨芙清

(北京大学计算机科学与技术系 北京 100871)

摘要 现有的软件复用技术通常是围绕着库来组织利用标准的和通用的可复用资源.例如,面向对象编程环境中的类库和通用构件库.然而,这种以库为中心的复用方式在一定程度上忽略了可复用资源的项目相关信息,而项目相关信息记录了可复用资源的应用语境(Application Context).应用语境有利于可复用资源的理解和使用.为此,提出了一种围绕着一个项目的文档来组织和利用可复用资源的复用途径,即以项目为中心的文档复用,用以支持在相同应用领域中一族软件的开发.探讨了当前的面向对象方法在支持文档复用方面的一些局限性,提出了一种增强文档的方案.针对文档中可复用资源的浏览和维护,探讨和提出了对在文档中导航的需求.

关键词 复用,面向对象,面向对象方法.

中图法分类号 TP311

软件复用是提高软件生产率和软件质量的有效途径.由于面向对象方法中的封装和继承机制,在软件开发中采用面向对象范型被普遍认为可以提高软件的可复用性,包括代码的可复用性以及分析和设计的可复用性.目前在学术界和工业界,大家的注意力大都放在标准的和通用的可复用资源的提取和利用上.例如,面向对象的编程环境中的通用类库,是在面向对象编程中复用代码的一个有效的手段.设计模式(design pattern)[1]和框架(framework)[2]的研究探讨了在面向对象软件设计中反复使用的设计知识的表示和利用,它们可以表示为由一组相关的类构成的结构,是一种面向对象的设计复用.这些通用的可复用资源,包括类、模式和框架等,通常是放在一个库中,并按照某种分类方法进行分类.软件开发人员围绕着库,浏览和查找可复用资源,并将资源应用在软件开发中.我们将这种复用方式称为以库为中心的软件复用.然而,这种以库为中心的复用支持在一定程度上忽略了可复用资源的项目相关信息,而项目相关信息记录了可复用资源的应用语境(application context),应用语境有利于可复用资源的理解和使用.

良好地组织软件文档,包括分析文档、设计文档、实现文档和测试文档等,可以有效地提高软件生产率和质量,并降低软件维护的开销.根据应用开发经验,我们发现一个应用项目的文档,特别是由复用专家开发的项目的文档,对在相同应用领域中类似应用开发还具有很高的复用价值.与库机制对比,一个软件项目的文档可以看成是另一种组织可复用资源的方式,其中包含着更多的可复用资源的应用语境.一个可复用资源的应用语境包括该资源与应用系统中其他部分的接口;该资源的使用是否还需要其他的可复用资源等等.此类信息可以用来理解和指导文档中的局部可复用资源(类和结构)的使用.此外,项目文档本身作为一个整体也是一种可复用资源.文档中的可复用资源的应用语境能对在相同应用领域中的应用开发项目提供更好的复用支持.因此,本文探讨了围绕着一个项目的所有文档来组织和利用可复用资源的技术,我们称之为以项目为中心的文档复用,强调应用项目的文档对相同应用领域中应用开发的复用支持.

* 本文研究得到国家自然科学基金和国家"九五"科技攻关项目基金资助.作者陈小群,1962年生,博士,主要研究领域为软件工程,面向对象技术,软件复用.邵维忠,1946年生,教授,主要研究领域为软件工程,面向对象方法和技术.梅宏,1963年生,博士后,教授,主要研究领域为软件工程,软件开发环境,构件技术,软件复用.杨芙清,女,1932年生,教授,博士生导师,中国科学院院士,主要研究领域为软件工程,软件开发环境,软件复用,构件技术.

本文通讯联系人:陈小群,北京 100871,北京大学计算机科学与技术系

本文1998-02-12收到原稿,1998-03-25收到修改稿

"以项目为中心"并不是一个新的概念,例如,现有的编程环境大都是以项目为中心的,即以应用系统的开发项目为管理单位,组织和管理开发人员的活动以及开发过程中的产物(文档、程序代码等).在应用程序的编写过程中,编程人员需要创建一个项目文件(project file),并基于该文件编辑、编译、调试、运行应用程序.

以项目为中心的软件复用途径就是通过项目的文档来组织和管理可复用资源,并利用文档中的语境信息理解和浏览这些资源,以便在相似的应用系统开发中复用它们.该途径的价值体现在以下几个方面:

(1) 通过例子学习.一个项目的所有文档提供了一个完整的实例.

(2) 支持相同应用领域中软件产品族的开发.同一领域中的两个软件产品,从需求到设计和实现存在大量的相同之处.

(3) 支持软件开发周期中各阶段的复用.不仅代码文档,分析和设计文档也是有效的可复用资源.

(4) 为可复用资源提供了充分的上下文相关信息.这些信息可用来理解和指导软件复用.可复用资源,比如对象类,其上下文相关信息包括:与系统中其他部分的接口;该类的修改对系统的影响等.

(5) 对领域分析[3]提供支持.领域分析需要对特定领域中的多个项目进行分析,提取通用的领域模型和软件的体系结构.项目文档是必不可少的素材.

大致上,我们可以将软件开发与棋类活动做个比较,一个软件项目对应于一盘棋.在下棋的时候,我们常常会用到各种定式,这些定式可以看作是软件复用中的通用可复用资源.然而,要成为下棋的高手只记住了一些定式是不够的,还必须学会灵活地运用它们.为此,棋手们的一项必不可少的功课就是研究高手的棋谱.这些棋谱可以看作是软件项目的文档.

作为一项工程活动,软件开发比下棋要复杂得多.有效的文档复用要求文档本身易于理解,在项目开发中采用复用技术,并在文档中显式地记录可复用资源,以及有效地从文档中获取可复用资源的手段.在文档中以良好的结构显式地表示可复用资源及相关信息可以有效地提高文档的可复用性.另一方面,如果不能高效和有效地从文档中获取可复用资源,文档复用的好处也会丧失掉.本文对这两个方面的问题进行了探讨.第1节介绍如何在面向对象的文档中显式地表示可复用资源.第2节探讨了为获取和维护文档中的可复用资源而需要的导航机制.第3节总结了文档复用的优点,并介绍了有关下一步工作的设想.

1 在软件项目的文档中表示可复用资源

软件项目文档中的信息包括图形和正文两个部分.图形方式适用于表示一些简单的信息,而复杂的细节需要用正文来表述.

软件项目的文档记录了软件的分析模型、设计模型、程序代码以及其他相关的信息,其中,分析文档记录分析模型,设计文档记录设计模型.这些文档又分为两个部分:应用系统的文档和可复用资源的文档.可复用资源的文档(如框架的文档)记录了可复用资源的模型、代码和相关信息.对于像设计模式和框架这样复杂的可复用资源,其文档可能包含大量的复杂内容,这些内容没有必要也不应该放在应用系统的文档中.例如,文献[1]给出的设计模式的描述模板包括问题、解、可用性、实现、例子等许多部分,如果把整个描述放在应用系统的文档中,将使得文档十分庞大和重复.

文档中的可复用资源包括构件类、设计模式、框架以及它们的模型表示.例如,构件类在设计模型中表示为类规约(specification),在程序中表示为代码.为了更好地支持文档(或模型)的复用,现有的面向对象方法应该增强在以下几方面的支持.

(1) OOA(object-oriented analysis)和OOD(object-oriented design)的区分

在OOA中,重点是捕获问题域的语义特性,建立需求模型;在OOD中,重点放在捕获求解域的语义特性,建立设计模型.这样建立的模型有利于分析复用和设计复用.某些面向对象方法[4]不对OOA和OOD加以严格的区分.

(2) 精确的语义描述

精确的语义描述有助于模型的理解.用当前的面向对象方法,如Coad-Yourdon[5,6]方法、OOAD(object-oriented analysis and design)[7]和OMT(object modeling techniques)[8]等建立的模型中,类及对象有良好的语义,

甚至可以用面向对象程序设计语言一一对应地描述它们.但是,对类及对象之间的关系却没有标准的语义和用法[9].当前的面向对象方法的符号体系不支持显式地表示多样的关系语义,通常是用一个简单的符号表示它们,所表达的概念范围太广而不能给出精确的含义.特别是在OOD中,绝大多数的面向对象设计方法并没有真正解释类及对象之间关系的含义,许多关系的语义是隐含的.例如,在OOD中这些方法不支持可见性*、拥有**和共享***等语义特性的显式表示.一般-特殊关系是个例外,它具有定义良好的语义规范.

(3) "为什么"的描述

了解设计这些关系的目的或用途有助于模型的理解.例如,在OOD模型中引入一个关系,其目的可能是为了分隔稳定部分和易变部分,或使一个对象可以访问另一个对象等.

(4) 设计知识的表示

一般的面向对象方法着重于应用系统的建模,较少支持建模过程中用到的知识的表示.面向对象的设计模式和框架是表示开发人员的经验和知识的两种有效的方式,在文档中标识并记录这些可复用资源是以项目为中心的软件复用的重要方面.

从以上分析我们可以看出,现有的面向对象方法对类和对象之间关系的描述比较简单,不利于文档和可复用资源的理解.另一方面,现有的方法缺乏对软件复用的支持,尤其缺乏对设计模式和框架的支持.为此,我们给出一个增强应用系统的文档的方案,以增强模型中关系和可复用资源的描述.表1列出了文档中增加的特性.

表1

应用系统的文档	关系	可复用资源
OOA文档	领域专有的术语 语义特性	构件类
OOD文档	语义特性 设计目的 包含关系	构件类 设计模式 框架
程序文档	包含关系	构件类的实现 设计模式的实现 框架的实现
文档之间	OOA模型与OOD模型之间的映射 OOD模型与程序之间的映射	

在OOA中,重点放在获取问题域的语义特性,构造需求模型.类及对象之间的关系的命名采用领域所特有的术语,而不是用像"实例连接到"、"是...的部分"这些一般性的术语,从而更准确地反映连接关系的问题域含义.例如,对车主与汽车之间的关系和旅馆与客房之间的关系,我们命名为"车主拥有汽车"和"旅馆有客房",而不是"车主连接到汽车"和"客房是旅馆的部分".此外,我们为模型中的关系增加一个语义特性项,用以进一步详细描述关系的语义特性.这些特性包括:暂时的(如房间-桌子)、元组-元素(如销售(买主,卖主,商品))、组-成员(如旅馆-客房)等.

分析模型中需要标识的可复用资源是构件类,这种构件类具有分析阶段的语义,用于静态地分析类模型的创建.

在OOD中,我们在关系的正文描述模板中添加语义特性项和设计目的项.语义特性项用以细化关系的设计语义,设计目的项描述为什么引入该关系.

设计模型中的可复用资源包括构件类、设计模式和框架,其中,框架可以看成是组织构件类和设计模式的一

---

* 对象的可见性确定了一个对象以什么形式进入一个方法的作用域.例如,对象可以作为方法的参数、局部变量、新创建的对象或当前执行该方法的对象的部分.

** 一个对象拥有另一个对象是指一个对象的删除将导致另一个对象的删除,两者有共同的生存期.

*** 一个对象,如果至少有两个对象引用它,则称该对象是共享的.

种机制,构件类和设计模式属于框架描述.一个框架可以包含多个构件类和多个设计模式.这种包含关系可以用来浏览和追踪可复用资源.

分析模型和设计模型中的可复用资源是以规约的形式来表示的.在程序代码文档分析模型中的可复用资源被表示为代码实现.与设计模型类似,在程序代码中框架与构件类和设计模式之间存在着包含关系.

OOA模型与OOD模型之间的映射关系记录了OOA模型中元素(包括可复用资源)与OOD模型中元素的对应关系,即OOD模型中哪些元素实现了OOA模型中哪些元素的语义.通过这种映射关系,我们可以从OOA模型跟踪到OOD模型中的对应元素,浏览和提取可复用资源,反之亦然.类似地,OOD模型与程序代码之间的对应关系有助于浏览OOD模型和程序代码中的可复用资源.

2 对导航的需求

文档复用效果的好坏,一方面取决于文档的组织,详细的语义、设计意图以及设计知识的显式表示;另一方面取决于浏览和定位可复用资源的便利程度.没有一个高效的浏览方法,文档复用的好处也就不存在了.面向对象模型是由类、对象和它们之间的关系组成的复杂的网状结构.在这样一个网状结构中,通过沿着关系导航来浏览面向对象模型,我们可以将注意力集中在相关的部分,找到完整的可复用资源.此外,当为了达到更有效的复用而改动模型时,例如,将子类的一些特性(即属性或服务)移到较一般的类,或用设计模式改造模型中的某些结构,采用导航方式可以使我们跟踪改动所影响到的部分,并相应地对这些部分进行修改.

面向对象模型通常是由大量的小实体,而不是少量的大实体组成的.要理解一个实体,可能需要追踪一串实体,以定位定义该实体的地方.对这样一个复杂的系统,明确一个实体与其他实体有几种语义关联,以及一个实体的改动会对其他实体产生何种影响,对定位和维护具有完整语义的可复用资源是必不可少的.同时,这对高效导航工具的设计有重要的指导作用.

2.1 类的特性的定义

类的特性,即属性和服务,有5种定义方式:(1) 在本类中定义;(2) 通过一般-特殊关系在超类中定义;(3) 通过整体-部分关系;(4) 通过消息连接关系;(5) 通过实例连接关系.通过关系定义特性有一些不同的情形,并且不一定是在直接相关的类中定义特性.例如,通过一般-特殊关系定义特性,一种情况是,特性在超类中定义,子类中不改变;另一种情况是,最初特性是在超类中定义而在子类中改变.特性可能在超类的超类中定义.因此,要得到一个类的完整信息,仅获取该类本身的定义可能是不够的.

2.2 模型的改变

模型的改变有以下几种情形:类特性的插入、删除和修改;类的插入和删除;类的合并;类的分解;关系的插入和删除.其中,类的合并和分解可以由类的插入和删除实现.

2.3 改变对模型的影响

类特性的插入、删除和修改,通过一般-特殊关系,可能影响该类与子类特性的语义一致性,例如,子类从该类继承的特性被修改.当发生不一致时必须分解该类或在子类中插入自定义的特性,使子类适应变化.通过消息连接关系,类中服务的改变可能影响消息连接到该类的类,即引用该类的服务的类.实例连接和整体-部分关系不影响类的封装性,因此,类的内部特性的改变一般不影响通过这两种关系关联的类.

子类特性的改变也可能影响到超类.例如,如果一个特性在所有子类中定义,则可以在超类中定义它,以避免冗余.

类特性的改变可能导致需要删除或插入某些关系.例如,一个类不再从其直接超类继承任何特性,在这种情况下需要删除该类与直接超类之间的一般-特殊关系,在模型中找到更一般的类,继承其特性.

类和关系的插入与删除需要特别当心,否则可能破坏模型的语义完整性.仅仅插入类不会对模型中的类产生影响,除非在插入的类与模型中的类之间插入关系.共享的类的删除会影响到多个类.具有拥有语义关联特性的类,其中一个类的删除导致其他类也必须删除.一个类只有当它没有子类的时候才能被删除.

3 结 论

以项目为中心的文档复用对在相同应用领域中一族软件的开发具有特别重要的意义,为通过实例学习的方式提供了基础.一个项目的文档提供了丰富的上下文相关信息,便于我们理解可复用资源是如何使用的.

文档复用效果的好坏有3个决定性因素:文档编写人员(包括分析人员、设计人员和编程人员)的经验、文档中显式记录的可复用资源的多少、浏览和提取可复用资源的工具的效率.本文论述了现有的面向对象方法在表示可复用资源方面的不足,以及导航对高效地浏览、提取和维护可复用资源的必要性.由于篇幅的限制,本文简要地介绍了一种表示可复用资源的方案.

目前,基于一个已经实现了的支持面向对象开发方法的工具,我们正在开发支持文档复用的原型系统.该原型利用工具所提供的建模设施,增加了一个导航工具,并利用数据库支持可复用信息的记录.

致谢 作者感谢王远宏同志、刘晓丹同志和陈兆良同志,感谢他们在原型实现中所做的工作.

参考文献

1 Gamma E *et al*. Design Patterns: Elements of Reusable Object-oriented Software. Reading, MA: Addison-Wesley, 1995
2 Pree W. Active guidance of framework development. Software Concepts and Tools, 1995,16(3):136~145
3 Ruben, Prieto-Diaz. Domain analysis: an introduction. Software Engineering Notes, 1990,15(2):47~54
4 Embley D *et al*. OO systems analysis: is it or isn't it. IEEE Software, 1995,12(7):19~33
5 Coad P, Yourdon E. Object-oriented Analysis. Englewood Cliffs, NJ: Yourdon Press. 1990
6 Coad P, Yourdon E. Object-oriented Design. Englewood Cliffs, NJ: Yourdon Press. 1991
7 Booch G. Object-oriented Design with Applications. Benjamin/Cummings, 1991
8 Rumbaugh J *et al*. Object-oriented Modeling and Design. Englewood Cliffs, NJ: Prentice-Hall, 1991
9 Civello F. Roles for composite objects in object-oriented analysis and design. ACM SIGPLAN Notices, 1993,28(10):376~393

Supporting Project-centered Object-oriented Reuse

CHEN Xiao-qun SHAO Wei-zhong MEI Hong YANG Fu-qing

(*Department of Computer Science and Technology Beijing University Beijing* 100871)

Abstract Current software reuse techniques usually utilize a repository to organize standard and generic reusable assets, such as class libraries and generic component repository in object-oriented programming environments. However, this repository-centered reuse ignores the project related information to some extent, which describes application context of reusable assets. Application context is useful to the understanding and utilization of the assets. In this paper, an alternative object-oriented reuse approach is presented, which utilizes the documentation of a project to organize reusable assets. This so-called project-centered documentation reuse is very useful to the development of a family of software in the same domain. In the paper, some limitations of current object-oriented methods for supporting documentation reuse are discussed and a new method for enhancing documentation for reuse is provided and analyzed. Then, aiming at browsing and maintaining reusable assets in documentation, the needs for navigation in documents are discussed and enumerated.

Key words Reuse, object-orientation, object-oriented method.