一种基于衍生类的动态类修改方法*

陆 陪 于大川 吕 建

(南京大学计算机软件新技术国家重点实验室 南京 210093)
(南京大学计算机软件研究所 南京 210093)
E-mail: lj@netra.nju.edu.cn

摘要 在面向对象语言的动态程序设计环境中,动态修改一个类时,会导致该类已经存在的那些活动对象难以处理的情况.本文提出了一种基于衍生类来实现类的动态修改的方法,并与其它方法进行了比较.

关键词 原型速成,面向对象,动态程序设计环境,动态修改,衍生类,源类.
中图法分类号 TP311

传统的软件开发的瀑布模型是由需求分析到系统设计、程序编码、调试连接、测试确认的自顶而下的过程.这一开发模型在软件开发领域长期处于主导地位,虽然对于很多软件系统的开发是行之有效的,但存在如下问题:(1) 用户的需求往往并不能在需求分析阶段就明确下来,而是随着整个软件项目的进行逐渐明确的.(2) 即使用户的需求在需求分析时已经明确,由于软件系统的设计、编码、测试人员往往不是同一个群体,一些特定的用户需求说明在经过不同人员的理解之后,与原有需求出入甚远.

在这种情况下,出现了原型速成软件开发模型.[1]在这种模型下,软件开发人员和用户紧密合作,利用原型速成工具快速开发出最终系统的一个原型,并在此原型的基础之上,确定用户需求.而后在此需求的基础上,进一步开发软件产品.原型速成技术主要有如下优点:(1) 使得用户需求可视化;(2) 为软件开发人员和用户提供了一个双方都可以参照的系统大致框架;(3) 使得软件开发过程处于一种循环的方式,软件的开发和测试更紧密地结合在一起.

随着面向对象技术的发展,面向对象语言被广泛地使用.在原型速成思想的指导下,出现了基于面向对象语言支持原型速成的动态程序设计环境.在这种环境中,程序员可以交互地输入对某些类进行测试的程序,例如创建这些类的对象,并向这些对象发送消息等等.在程序的执行过程中,程序员可以对涉及到该程序的任何一个类以及任何一个活动对象的状态和结构进行修改.修改之后,继续以交互的方式运行原程序.由此不断地修改、测试、完善用户类,从而迅速生成这些类的原型.这种可以在程序运行时刻进行修改的类称为动态类.在程序运行时刻对它们进行的修改称为动态修改.一个动态类其实就是一个原型类.

这种用动态类修改来支持原型速成的方法的一个明显的问题是,当修改了一个已存在活动对象的类时,会使得那些活动对象和更改后的类不一致.特别地,当改动了类的成员变量描述部分,会直接导致那些活动对象的存储结构不符合新类的描述,处于一种"无类可依"的状态,因而会使得对那些对象的操作不合法,导致程序执行的紊乱.现有的动态程序设计环境的解决方案,包括 Smalltalk 环境的将对象进行转化的方法,都不能很好地解决这一问题.

针对这一情况,本文以我们与美国 Motorola 公司的合作研究项目"面向对象软件开发环境 MagicFrame"为背景,提出了一种基于"衍生类"的动态类修改方法.其基本思想是,在对某一个类进行动态修改时,先以该类为源类生成它的衍生类,修改直接作用到衍生类上,而并不是该类.这样可以有效地排除"无类对象"的出现,并且还可以直观地比较分属不同类的对象的行为,以验证修改结果,更好地支持类的演化完善过程.

1 面向对象的动态程序设计环境

所谓面向对象的动态程序设计环境,是指基于面向对象语言来支持动态程序设计的开发环境.对面向对象语言而

* 本文研究得到国家杰出青年科学基金、跨世纪优秀人才计划基金和国家攀登计划基金资助.作者陆陪,1974年生,硕士生,主要研究领域为面向对象语言与环境,并行程序形式化方法.于大川,1977年生,硕士生,主要研究领域为面向对象语言与环境,并行程序形式化方法.吕建,1960年生,博士,教授,博士导师,主要研究领域为软件自动化,面向对象语言与环境,并行程序形式化方法.

本文通讯联系人:陆陪,南京 210093,南京大学计算机软件研究所

本文 1997-02-03 收到原稿,1997-04-22 收到修改稿

言,动态程序设计指的是,在程序运行时刻,程序员可以对当前时刻所有活动对象和组成该程序的类进行监控和修改,而修改之后可以继续运行该程序.Smalltalk-80 程序设计环境[2]就是一个经典的面向对象动态程序设计环境.

在我们设计的动态程序设计环境中,程序是由程序员交互地输入到工作平台(Workspace)中去并执行的,在这个交互的过程中,程序员可以用对象监控器(Inspector)来查看和修改活动对象;用类层次树浏览器(Class Browser)来查看和修改类层次树(其中包括系统类和用户类);用类设计器(Class Designer)来设计、查看和修改一个类;用调试器(Debugger)来对程序进行调试.整个系统的框架如图1所示.

图1 动态程序设计环境示意图

动态程序设计环境是基于原型速成思想基础之上的.对面向对象语言而言,动态可修改的类就是一个个的"类原型",由于对它们的修改可以在运行时刻进行而无需重新编译,并且修改的效果实时可见,所以可以有效地用来对类进行快速的开发.

由于对类可以动态修改,动态程序设计环境下程序是通过解释的手段来执行的.显然,其执行效率是比较低的.并且,由于没有编译执行方式下的全局的类型检查机制,它是类型不安全的.

在对类进行动态修改时,一个显而易见的问题就是,假如一个类已经存在活动对象,对它进行修改,原有那些对象如何处理?对类的修改分两种:修改成员变量部分和修改成员函数部分.对成员函数部分的修改,也就是改变了对象的行为,对已经存在的对象的实际存储结构没有直接的影响.但是,对成员变量部分的修改,会直接导致这些对象存储结构和新类的描述不一致.由此可能产生一系列诸如类型错误的问题,并使得继续执行下去的程序处于一种逻辑上紊乱的状态.

对这一问题过去常用的两种解决方法是:(1)丢弃所有的这一类对象,这显然是一种保守的方法.由此的确可以保证程序中每一个对象都和其所属的类的描述一致.但是,这样一来就相当于重新启动程序执行,继续执行就失去了其本来意义,从而动态对类进行修改也就失去了其本来的作用.(2)根据原有的那些对象的内容,将它们全部转化成为符合新类描述的新的对象.这显然是一种乐观的想法.基于Smalltalk语言的动态程序设计环境VisualAge[3]就是这样处理的.由于Smalltalk语言是一个无类型的语言,它这样处理相对容易,但仍然大大地降低了动态修改的执行效率.对一个有类型的面向对象语言的类可作各种各样的修改,不但可以增加、删除一个成员变量,还可以改变其成员变量的类型.要将它们进行转化要复杂得多,对执行效率的影响也更大.

2 基于衍生类的动态类修改方法

2.1 衍生类和源类

针对上述问题,我们引入衍生类(Cloned Class)和源类(Source Class)的概念.

衍生类就是,动态程序设计环境中由一个类复制出来的用于作动态修改的类.被复制的类称为源类.由源类生成衍生类的过程称为衍生(Clone),我们这里所指的衍生类和衍类[4]是两个完全不同的概念.衍生类是我们为了动态修改的需要在程序运行时刻创建的类,完全是动态时刻才存在的.而衍类就是通过类间继承得到的子类,是一个静态时刻的概念.

在动态程序设计环境中,当对一个类 A 进行动态修改的时候,系统就自动生成该类的一个衍生类 A'.所有的修改直接作用于该衍生类 A',而不是类 A.如图2所示.当修改完毕后继续运行程序时,源类 A 已经存在的对象仍然属于类 A,其行为不受任何影响;而当需要创建类名为 A 的新的对象时,系统自动生成其衍生类 A' 的对象,而不是源类 A 的对象.而后,根据对象的分属的类的不同,进行不同的操作.这时,程序中虽然只有同一个类名 A,但实际上存在两个不同的类 A 和 A' 的对象,由系统隐式地自动加以区分.

2.2 衍生类树

由于对一个类的修改可以从多个侧面并行地进行,因此对同一个类而言,可以同时生成其多个衍生类.并且,由于在已经进行修改的衍生类基础之上,还可以进一步修改,因此,对一个类 A 进行的修改可以得到一棵以 A 为根节点的由衍生类构成的树,树中每一个节点就是一个相对于其父节点的衍生类(也就是其祖先节点的间接衍生类),每一根树枝就是一个动态修改过程.我们称之为类 A 的衍生类树或动态修改树.如图3所示.

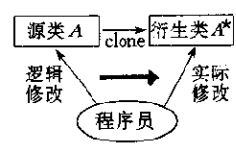

图2 动态修改衍生类示意图

类 A 动态修改的过程,就是以 A 为源类根节点的衍生类树的扩展过程.在这棵树上最新生成的节点 A_i 称为当

前衍生类,也就是最近作了修改的类.在此状态下继续程序的执行时,若要生成类名为 A 的对象,实际上系统负责生成当前衍生类 A_i 的对象 a.而后 a 一直作为 A_i 的对象来处理.我们将当前衍生类的对象生成称为该类的有效化.

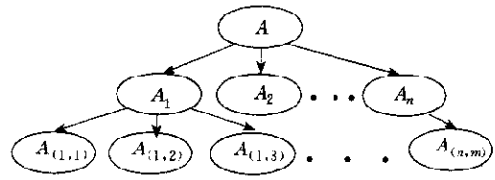

图3　由衍生类构成的动态修改树

假设衍生类树的当前衍生类为 B,在 B 有效化之前,所有对 B 所进行的修改直接在 B 的基础上进行,无需生成 B 的衍生类.新的修改和原来 B 上已经进行的修改合并作为一次修改,也就是作为衍生类树上的一根树枝.这样规定的原因是显然的,因为如果类 B 在有效化之前发生改动,那么它就不可能作为一个类来实例化其对象了,也就无需在衍生类树中保存它.在这样的规定下,不难看出,衍生类树中的所有非叶节点类都生成过其对象.

程序员认为对类的修改已经很成熟时,可以将源类更新为它的衍生类树中的某一个衍生类(此时若仍有源类的活动对象存在,由系统删除它们),并释放该衍生类树,从而完成一个动态修改过程.这一过程称为更新确认.一般而言,动态类的更新确认都是在程序执行完毕的时候进行.

值得注意的是,在绝大多数情况下,动态程序设计环境中被动态修改的类,作为处于开发过程中的不成熟的原型类,都是类层次树中的叶节点类(不再有子类).因此我们规定.对类 A 动态修改时,如果类 A 有子类,则它们仍然作为源类 A 的子类,而不是转变为衍生类 A^* 的子类.这样会使得对 A 上的动态修改无法立即作用于 A 的子类上,如果需要对类 A 进行修改的语义传递到其子类上,则可以在动态修改该类之后立即进行更新确认,或直接采用静态修改的方法.

2.3 优　点

与前文所述的删除被改类活动对象和 VisualAge 环境的转化被改类活动对象两种直接类修改方法相比,采用衍生类来进行动态修改的方法有如下优点:

(1) 对衍生类的修改不影响源类,不会出现源类对象"无类可依"的情况,也就无需删除或对原有对象进行繁琐的转化.程序的继续执行在逻辑上仍然清晰、合理.

(2) 在动态程序设计环境中,对类的修改并不是一步到底的,而常常是具有反复性的活动.一次修改操作常常在下一次修改中被恢复.如果采用直接修改的方法,就会有一个反复的处理过程,大大地降低了执行效率.而通过衍生类的方法,对已经进行的错误修改无需任何处理,只要在源类更新确认时保留原状即可.

(3) 衍生类带来的另一个显著的好处就是,由于源类的对象可以和衍生类的对象并存,所以可以对动态修改的效果进行直观的比较.例如,已有一个窗口类 W 的对象,即一个窗口 w_1.当对它的某一种界面效果不满意时,可以修改类 W,此时实际上被修改的是类 W 的衍生类 W^*,修改之后,程序员可以立即生成 W^* 的一个对象 w_2.这样,程序员可以通过对两个不同的窗口对象 w_1 和 w_2 直观地进行比较,从而决定是否需要作某种修改.使用衍生类的方法所带来的这一点好处是前文所述的两种处理方法所不能提供的,而这一点恰恰充分发挥了动态修改的灵活性,从而更有利于快速原型的生成.

衍生类实质上就是计算机领域常用的缓冲技术在动态修改中的应用.并且,它扩展了缓冲的思想,一个类可以有一棵"缓冲树".因而,它不但提高了动态修改的安全性,还使得动态修改具有更大的灵活性.

2.4 实　现

基于衍生类的动态类修改方法带来的好处是以其相对复杂的实现为代价的.

由于衍生类不同于一般的类,而是基于某一个源类进行修改得到的新类,所以,它的类存储结构不同于普通的类.它的类结构中分两种成员:(1) 复制成员,即源类中未经修改的成员;(2) 变动成员,即源类中经过改动的成员.当然,也包括新增加的成员.衍生类中的所有成员都用指针的方式加以实现.如果是复制成员,就直接指向源类的相应成员.如果是变动成员,就指向该衍生类空间的相应成员.

每当开始对程序某一个类 A 进行动态修改时,系统就为该类创建一棵衍生类树,树中每一个节点就是一个衍生类结构.最初时只有一个节点 A.进行第 1 次修改时,系统首先生成该类的一个衍生类 A_1,并给它起一个类名,表明它是类 A 的衍生类,且保证它不与任何其它用户程序中的类有冲突.并将它作为类 A 的当前衍生类.所有的修改都作用于 A_1.修改完成后,A 的衍生类树上就多了一个 A_1 节点.继续执行程序时,类 A 的原有已经存在的对象仍然按 A 来解释其行为,而再遇到生成类 A 的对象 a_1 之类的语句时,则解释为生成类 A 的当前衍生类 A_1 的对象 a_1,并且系统标记 a_1 为类 A_1 的对象,标记 A_1 为有效化的类.以后对 a_1 的操作均按类 A_1 来进行.

当程序员需要进一步进行动态修改时,系统以可视化的方式提示给程序员修改类 A 的衍生类树,程序员可以选

择其中的任何一个类 A' 进行修改. 如果 A' 没有有效化,则所有的修改直接作用于 A',不再以 A' 为源类生成新的衍生类节点. 如果 A' 是有效化的类,则以 A' 为源类生成其衍生类 A'',并作为新的当前衍生类. 这样,随着修改的进行,类 A 的衍生类树不断地扩大,程序中由衍生类生成的对象越来越多.

当程序执行完毕时,系统以可视化的方式提示给程序员类 A 的衍生类树,并要求从中选择一个来替换 A,也就是更新确认的过程. 当然,程序员完全可以在动态修改进行的过程之中就进行确认,这就相当于直接修改的情形,此时若仍然存在已被确认修改的源类的活动对象,则由系统来删除这些对象.

在实际情况下,一个类的衍生类树的深度通常不会超过 3 层,树中所有节点通常不超过 5 个. 动态修改时,一般都是改动类的很小的一部分成员,并且源类和衍生类的复制成员以共享方式实现,因此,维护一棵衍生类树的空间和时间开销并不大. 程序执行时,也只是在生成对象的时候生成当前衍生类的实例,并标记该对象所属的类为有效化的类,因此,在解释执行程序时也几乎没有什么额外的时间开销.

2.5 对象结构的动态修改

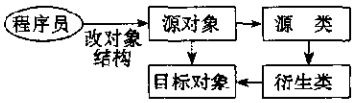

图4 对象结构动态修改示意图

在动态程序设计环境中,不仅可以动态地来修改类,还可以动态地来修改当前程序运行的活动对象. 对对象的修改可以分为两个方面来看. (1) 对象值的修改,即修改对象的数据成员的值,相对较为简单;(2) 对象结构的修改,即修改对象的数据成员的类型. 显然,对象结构的修改其实就是通过对象来间接地修改对象所属的类.

同样,我们可以用衍生类的方法来支持对象结构的修改. 当程序员修改某一个当前活动对象的结构时,我们以其所属的类为源类,按照在对象上作的相应的修改,生成衍生类,而将被修改的对象作为新生成的衍生类的实例对象. 也就是说,每当修改对象结构时,我们总是先修改该对象所属的类. 这一过程如图 4 所示.

3 结束语

基于衍生类的动态类修改方法是我们在面向对象的动态程序设计环境 MagicFrame 的设计过程中,针对现有问题提出的一种解决方法.

虽然衍生类方法在某种意义下解决了动态类修改问题,能够保证对象和其所属类的一致性,但它并不保证动态修改的类型安全性. 实际上,动态修改的灵活性正是以牺牲其类型安全性为代价的. 在 MagicFrame 系统中,类型安全性问题是通过"冻结"机制来解决的. 当完成一个动态类的开发时,使用"冻结"机制,对其进行全面的类型检查,转化成为类型相对安全的静态类.

致谢 本研究与徐永森教授领导的 MagicFrame 课题组全体成员的合作和帮助是分不开的,在此衷心地表示感谢.

参考文献

1 Connell J L, Shafer L B. Structured rapid prototyping: an evolutionary approach to software development. Englewood Cliffs, N J: Yourdon Press/Prentice Hall, 1989
2 Adele Goldberg. SmallTalk-80: the interactive programming environment. Addison-Wesley, 1983
3 OTU School One, Smalltalk Developer/Tester Training Student Notebook. IBM Corp, 1995
4 Xu Jia-fu. Object-oriented programming language. Nanjing: Nanjing University Publication Ltd., 1992

A Method for Modifying Dynamically Classes Based on Cloned Classes

LU Pei YU Da-chuan LÜ Jian

(*State Key Laboratory for Novel Software Technology Nanjing University Nanjing* 210093)
(*Institute of Computer Software Nanjing University Nanjing* 210093)

Abstract In the object-oriented dynamic programming environment, dynamic modification of a class will cause a problem that active objects of the class are difficult to handle. In this paper, a method for modifying dynamically classes based on cloned classes is proposed, and the comparison with other methods is given.

Key words Rapid prototyping, object-oriented, dynamic programming environment, dynamic modification, cloned class, source class.

Class number TP311