基于 UML 的面向对象设计研究*)

Research about Object-oriented Design Based on UML

潘家志　强保华　余建桥
(西南农业大学信息学院　重庆400716)

Abstract In this article we analyze some characters of Object-oriented design, also we discuss the modeling methods of UML. We draw a conclusion that in system design process we should combine the techniques of Object-oriented and UML, so we can overcome the problems we encountered when using traditional developing methods. By showing some examples, we demonstrate that this method may be applied to crack the problems reside in traditional software developing methods.

Keywords UML, Object-oriented design, Modeling method

1 引言

瀑布模型曾经是使用得最为广泛的软件生命周期模型，基于该模型的结构化分析与设计是软件技术发展历史上的重要成就。但是，随着人们对应用需求的升级及技术的发展，基于这种模型的开发方法暴露出越来越多的弊端。主要表现在：

不便于参与开发的各类人员之间的交流；

在开发过程中缺乏反馈环节和循环过程，整个系统作为一个整体在运动，从系统建模与系统需求的收集，经过分析、设计、编码、测试，到系统配置运行，用户在产品开发结束之前见不到任何可以实际运行的东西；

随着规模的扩大，功能分解变得越来越困难，难以保证开发进度和开发的最后成功，产品质量无法保证。

面向对象的开发方法是继结构化分析和设计方法之后的又一个革新，它解决了结构化技术存在的两个问题。一个是过程和数据的分离以及需求分析模型和软件设计模型的不匹配，另一个是信息与实时系统开发方法的分离。统一建模语言(Unified Modeling Language，UML)是一种图形化的建模语言，它不仅支持面向对象的分析和设计，而且支持从需求分析开始的软件开发全过程。下面的研究将显示二者的结合会产生一些有助于解决上述问题的优良特性。

2 面向对象设计与 UML 建模概述

2.1 为什么要使用面向对象

面向对象是许多人历经多年研究积累的产物。面向对象的研究起源于60年代，但直到90年代，面向对象技术才趋于成熟。使用面向对象可以获得以下益处：

·面向对象的方法在生命周期的早期就将过程和数据研究融合在一起。

·面向对象方法使设计者将软件中的棘手问题利用封装特性隐藏起来，这些问题包括：复杂的数据结构和组合逻辑、详细的过程和数据之间的关系、高深的算法和设备驱动程序。

·面向对象方法可以提高系统质量。面向对象通过在类的级别上而不是在各子程序级别上提高代码重用来改进软件的可重用性。面向对象代码采用类的不变式确信的断言，借助自身进行验证，虽然不能证明代码绝对正确，但可以使代码行为的检查变得更容易，因此能提高可靠性。

·许多现代的面向对象语言和环境都支持错误检测和处理功能，因此有利于开发健壮的软件。获得健壮的面向对象代码的有效方法是将推断和恒定条件的概念与异常处理相结合。

·面向对象方法可以获得较好的可扩展性，能够保证用户的一些小的修改不会导致系统的灾难性后果。

·面向对象是支持图形用户界面 GUI 的最佳途径，而好的图形用户界面增加了软件的易用性。

2.2 UML 建模的特点

UML 已经被对象管理组织 OMG 所接受，作为软件系统建模的标准的可视化建模语言。UML 用各种图来创建模型，它是在三大面向对象方法学的基础上，抽象出的模型语言，并汲取了其它面向对象开发方法和近三十年软件工程的经验和成果。UML 是工具和方法无关的，是面向对象开发中的一种通用、统一、图形化模型语言。UML 所提供的可视元素构件可以设计、表达复杂的面向对象软件的体系结构。UML 的建模过程有三个基本特性：用例驱动、以体系结构为中心、迭代式的增量开发。

用例驱动　在 UML 建模中，通过用例来获取系统的所有的功能需求，驱动软件的开发过程。同时，用例是审核和测试系统的依据，它将应用于系统开发的所有阶段。

以体系结构为中心　使用 UML 来进行系统建模，必须尽早建立一个良好的系统体系结构，然后建立原型和进行评估，并在开发过程中不断细化。系统结构把系统划分为多个子系统，描述各个子系统的相互关系、相互作用、通信机制以及修改原则。

迭代式的增量开发　使用 UML 建模时，不应试图一次定义出模型的所有细节，而应分成一系列小的迭代过程。在每次迭代中都会重复所有的开发过程，只是每次迭代的重点不同。通过一次次的迭代，系统的功能逐步增加和完善，直到最后满足系统的需求定义，通过系统测试为止。

3 面向对象设计与 UML 集成

以下的几个面向对象设计的例子将全部采用 UML 的语法描述，讨论了面向对象领域的一些重要原则和准则。显示了在面向对象设计中结合 UML 的巨大优越性。

3.1 使用 UML 提高了类层次结构的清晰性

*)得到重庆市教委科技项目资助(No. 011806)。潘家志　硕士研究生，主要从事软件工程、数据库研究。强保华　硕士研究生，主要从事数据库、网络研究。余建桥　教授、硕士生导师，主要从事人工智能、数据库及软件工程研究。

继承是面向对象的基本特征之一，是一个如同 goto 结构一样功能强大、易于被滥用的结构。考虑如下的一个关于生物分类的例子，就可明白。为了表明熊猫 Panda 是一种熊 Bear，而且 Panda 是一种濒危物种 EndangeredSpecies；因此设计者可能给出如图1的类层次图。仔细分析一下，可知它是一个错误的设计。因为 EndangeredSpecies 可能包括一些和 Panda 无任何关系的物种，因此，Panda 不能从 EndangeredSpecies 类继承而来，尽管 Panda 可以从 Bear 继承。为了表述这些事实，可以先将图1分解为图2的样子。从图2中可以看到，EndangeredSpecies 和 NonEndangeredSpecies 是没有公共元素的，同时也是对其父类 Species 的完全划分。即一个物种要么是濒危物种，要么不是濒危物种。也没有一个物种既是濒危物种，又是非濒危物种。UML 的标准图符可以让用户加上一些文本信息，以说明类之间的关系。图2由两个分离的部分组成，并没有包含 Panda 是一个濒危物种的事实，因此，应该设计一种方法将图中的两部分联系起来。一种方法是在 Bear 类中增加一个实例属性 isEndangered：Boolean。尽管这种方法是可行的，但是，它牺牲了类的一般性的特征。作为修改的结构，如图3，给 Bear 的超类 Animal 增加一个属性 species，用来将它与 Species 对象关联。Animal 的每一子类代表一个物种，且属性 species 为该子类对象保存一个合适的固定值。许多的 Animal 类的聚集，就构成了 Species 类。在 Bear 类的对象中，无论 Bear.species 采用哪种方法获取其值，类 Bear 的对象均可在运行时通过如下代码找到其所属物种及物种相关特征：

```
self.species.isEndangered
self.species.maxWeight;
```

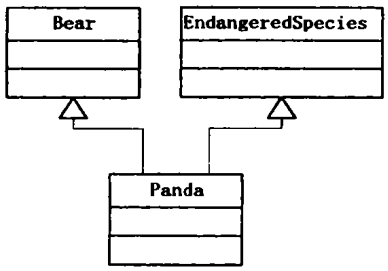

图 1　熊猫的继承关系

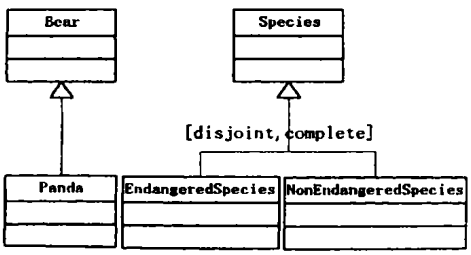

图 2　分离的类结构

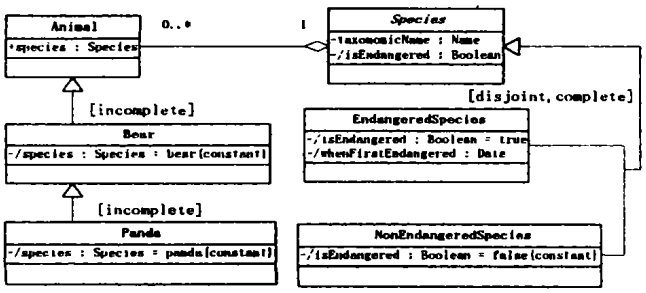

图 3　正确的继承关系

通过这个例子，我们得出一个重要原则：类的层次结构要非常清晰，要避免错误的继承关系。而 UML 的表示方法可以让用户直观地表达类之间的关系，而且既包含图形信息，又包含文字说明，能够十分清晰地表达设计意图。较之于其他的类的表示方法有巨大的优越性。

3.2　UML 对面向对象设计方法的支持

UML 建模可以对软件开发提供全程支持，而不光是在设计分析阶段。下面的研究将显示应用 UML 可以提高设计质量，利用它的支持功能，可以提高开发速度。我们以如下一个应用为例子。有个公司需要按照用户的喜好给每个顾客发送发票。有传真方式、电子邮件方式以及邮递方式。最开始的类设计可能如图4所示。可发送发票类 SendableInvoice 继承原始类 Invoice，在 SendableInvoice 中分解出诸如投递电子发票 emailInvoice 和传真发票 faxInvoice 之类的操作。在发票类 Invoice 中可以有理想的内聚结构。如果要创建一个新的发票，初始化一个可发送发票对象实例就行了，其中的传真发票操作 faxInvoice 能自动运行传真调制解调器，并能访问使用传真的发票信息。但是，当设计好这个类时，才发现它的使用范围太小了。如果要传真的不是发票信息，而是诸如通知、贺信之类的东西时，这种设计就不好使用了。如果引入混合类，将它修改成如图5的类结构，这个问题就解决了。在改进的设计中，使用了一个混合类 Sendable Document 可发送文档，它可以解决发送传真和电子邮件的问题。更重要的是，它不需要有关于发票的任何知识，是一个通用类，能传真和用电子邮件发送任何文档。需要创建一个表示新发票的对象时，调用 SendableInvoice.New 初始化该对象，对象名为 SendableInv，给出发票的物品信息，并与相关的客户 Customer 对象连接起来，就可以使用任何方式发送发票了。因为 SendableInv 继承了 SendableDocument，所以具有它的通信能力。发送由 SendableInv 产生的发票时，完成以下两个步骤即可。

(1)调用 SendableInv.createDoct，该操作创建能传真、能作为电子邮件发送或能打印的标准文档。在 SendableDocument 中定义表示该文档的属性，该文档的操作，也是在 SendableDocument 中定义。SendableInv.createDoct 的伪码如下：

```
public operation createDoct
begin
    self.clearDoct;
    get the invoice header;
    convert it to the text form headerText;
    self.appendTextToDoct(headerText);
    repeat
    get the next invoice line;
    convert it to the text form line text;
    self.appendTextDoct(lineText);
    until no more invoice lines;
end createDoct;
```

(2)现在已经将发票信息填入属性 doct 中了，需要发送出去，调用操作 SendableInvoice.sendToCustomer 完成，伪码如下：

```
public operation sendToCustomer
begin
    cust:Customer;=self.responsibleCust;
    case cust.prefCommMedium "MAIL": self.mailDoct (cust.
        name,cust.address);
"EMAIL":self.eMailDoct(cust.name,cust.eAddress);
"FAX":self.faxDoct(cust.name,cust.faxNumber);
    else…;
    end case;
end sendToCustomer;
```

当类结构设计通过检查，确认之后，可以使用 UML 的代码工具，进行设计的检查和代码框架的生成，这样用户就可以专注于类的实现之中，而不必关心各个类之间的复杂的接口问题。以下给出了图5中的类 SendableInvoice 的说明文件：

```
//Static Model
```

```
#ifndef_Top_Package_Package2_SendableInvoice_
#define_Top_Package_Package2_SendableInvoice_
//Include files
#include"Invoice.h"
#include"SendableDocument.h"
namespace Package2
{
class SendableInvoice:public Invoice,public SendableDocument
{
public:
        SendableInvoice();
        ~SendableInvoice();
        int createInvoice();
        int mailInvoice(String name,String address);
        int eMailInvoice(String name,String e_address);
        int faxInvoice(String name,String fax_number);
protected:
private:
};
//END CLASS DEFINITION SendableInvoice
}//Package2
#endif//_Top_Package_Package2_SendableInvoice_
```

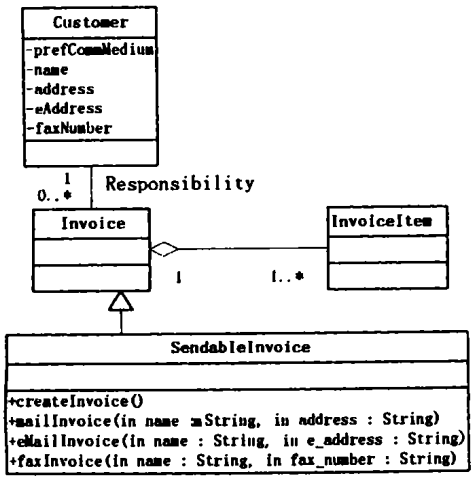

图 4 具有发送能力的发票类

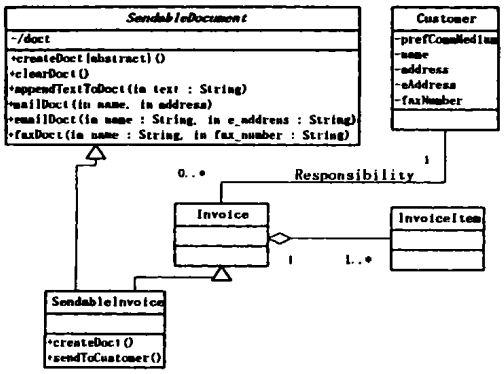

图 5 带混合类的层次结构

从以上的类的说明文件的框架中可以看到，UML 自动生成的代码能够表达类的设计，即 SendableInvoice 类由 Invoice 类和 SendableDocument 类通过公有继承而得到。

通过这些研究，我们可以得出一个结论：在不破坏类的内聚的前提下，采用混合类来增强类的功能，提高可复用性。UML 的支持可以使类的设计质量得到提高，它可以生成多种模型的汇总报告以及其他各类的反馈信息。此外，代码框架的生成能力有助于提高开发速度。

结论 通过以上研究，可知 UML 对面向对象业界具有的重要意义。虽然 UML 独立于语言和方法学，但支持所有的方法学。它覆盖了最重要的对象分析和设计的概念和相关方法。UML 是一种广谱的建模语言，适用于所有类型的系统建模，如实时系统、客户/服务器系统等。通过使用 UML 建模，它的图形化的元素易于理解、便于交流，同时还具有很强的描述定义能力，通过在早期建立原型和迭代式的增量开发，可以分解软件项目的风险，为开发过程提供多级反馈和多次循环过程。本文研究显示了在面向对象的设计中使用 UML 方法后，可以产生结构更清晰的类层次结构，更能够说明面向对象的一些抽象原则，有助于解决传统的面向过程方法所面临的各种问题。但是，如何通过 UML 的支持，产生可以评价类层次结构优劣、可以提供修改建议的辅助工具，如何将诸多的面向对象的原则集成到 CASE 工具中，都是值得我们进一步研究的问题。

参 考 文 献

1 Hatley, Hruschka P. Process for System Architecture and Requirements Engineering. New York: Dorset House Publishing, 2000
2 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Mass.: Addison Wesley, 1999
3 Booch G, et al. The Unified Modeling Language User Guide. Addison-Wesley Longman, Inc, Oct. 1998
4 Rational Software Corp. UML Summary V1.1. Sep. 1997
5 Rational Objectory Process-Introduction. Rational Corp. Press, 1997
6 林扬帆，李师贤. UML 分析模型中功能点计算的探讨. 计算机科学，2001，28(8)：85～88
7 王云，周伯生. 基于 UML 集成化支持环境的柔性软件开发过程. 计算机科学，1999，26(10)：73～77
8 邵维忠，梅宏. 统一建模语言 UML 评述. 计算机研究与发展，1999，36(4)：385～394

(上接第109页)

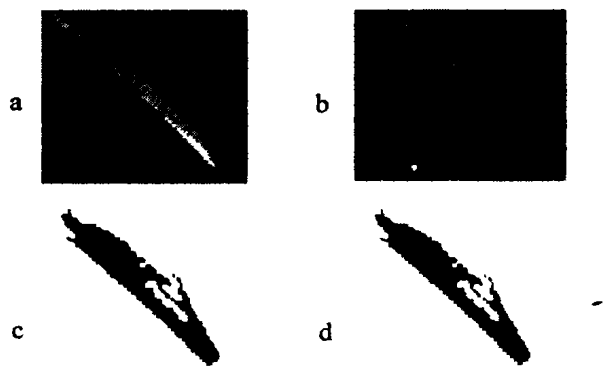

图5

边界跟踪不仅可以对任何连通成分进行标记，而且可以给定阈值参数来对灰阶图像中的任何区域进行边缘跟踪、标记，在图像处理中有着广泛的应用。

参 考 文 献

1 Hoang K, Nachimuthu A. Image processing techniques for leather hide ranking in the footwear industry. Machine Vision and Application Journal, 1996, 9: 119～129
2 Wilder J. Finding and evaluating defects in glass. In: Freeman H, Rummel P, eds. Machine Vision for Inspection and Measurement, Academic Press, 1989. 237～255
3 Jain R C, Kasturi R, Schunk B. Machine vision, McGraw-Hill Inc, New York, 1995
4 Rogers D F, 梁友栋，等译. 计算机图形学的算法基础. 科学出版社，1987. 78～83