

面向对象和面向过程的系统开发方法比较(上)

郭江 廖越虹 (北京航空航天大学软件工程研究所)

摘要: 本文为了讨论这一改变对系统开发现状的影响,对面向对象的开发方法和面向对象与过程相结合的开发方法进行了比较。在面向对象的开发中最显著的特点是它依赖于商业信息系统结构,它是完整商业企业的一个实体关系模型。特别是在应用系统开发中要考虑这个商业模型所需的变化。

本文分析了传统数据处理和面向对象计算之间的区别,以及这些区别对系统开发的影响。结合一些面向对象技术的结构化设计方法与纯粹的面向对象的系统设计途径是截然不同的。本文描述了面向对象的方法在系统开发生命周期各阶段的含义,并指出了这种方法在生命周期早期阶段的优势。尤其是用于系统开发早期阶段的CASE工具用面向对象的方法在这些阶段中可以达到较高的生产率和较好的质量保证。

在分析用于系统开发的面向对象的含义之前,先回顾一下面向对象的计算基础。本文只考虑那些与系统开发相关的概念,而忽略掉程序设计上的技术含义。

1. 面向对象的基础和系统设计

面向对象的系统使用执行方法的对象来模拟商业行动。尽管对象能执行与传统的输入-处理-输出过程相同的操作,但对象执行一个方法与过程调用仍有一些差别,其区别在于对结果的判定上。在调用一个过程时,这个过程势必遵循一种模式。比如一个数学函数,对于一个给定值的独立变量,与它相关的变量值是不变的。然而,一个对象对请求的响应依赖于请求发出时它的状态。这个状态可用对象的属性值来定义(即,变量),当一个对象和其它对象相互作用时,它的属性值可发生变化。

下面的例子说明了一个对象的响应是如何依赖于其变量状态的。一个发送消息的系统是以对象为基础的,这些对象可响应下列请求:创建、发送和显示消息及对象的建立和删除。一用户发出命令创建并把消息发送给另一用户,他就可以通过发出一个显示消息的请求看到这条消息。如果没有生成或传送消息,那么显示消息的命

令也就显示不出消息了。因此说,对显示请求的响应依赖于消息对象的状态。

使用对象来模拟商业活动就会使系统的定义和组织不同于使用传统的输入-处理-输出的方法。面向对象的方法依照可变数据来组织对象。相反,输入-处理-输出方法依照使用它的过程来组织数据。

这些区别从根本上改变了定义和规划计算机应用系统的方法。从面向对象的观点上来说,是把应用系统看作一组状态变化着的对象,而不是处理数据的过程。当把一个系统定义为一个对象集合时,所有相关的对象也同时进行了定义。例如,一个存货单系统包括存货项目,存货地点和存货补充。一个顾客发票系统包括顾客、销售额和存货项目。在存货单系统定义中,一个存货项目如果存放在某个地点,那么它就和存货地点有关。一个存货补充对象和存放在某个地点的项目通过改变存放于该地点的货物相互作用。

如果一个对象和多个应用系统相关,那么这个对象就应包含在所有需要它的系统定义中。例如,卖给某顾客的销售额与一个存货单控制系统相关,而它又与顾客发票系统密切相关,所以它既包含在存货单控制系统,也包含在顾客发票系统中。用输入-处理-输出方法来规划同样的系统时,包含在存货单和顾客发票系统中的销售额数据就会产生冗余和不协调的问题。而采用面向对象的方法时,两个系统都调用同一个销售额对象,就能避免这样的问题。

(1) 商业领域的信息结构。使用面向对象的方法来进行系统规划时,就需要所有相关商业实体和活动的信息。用这些信息就可以定义实体和关系,从而决定对象;

EMS 命令”和“当前用户”数据存储象征用户界面实体。另外两个实体是从数据流图得来的,包含了三个过程的“用户目录”和包含了“访问队列”过程与“用户队列索引”数据存储的消息中心。一般而言,数据流图的抽象分析一直持续到把所有过程和数据存储同实体联系起来为止。

(4)构造系统实体图。在第四步,要构造一个系统的实体图(见图 3)。这张图包括通过抽象分析定义的实体—用户界面,用户目录和消息中心,还有一个最高级的实体—电子邮件系统,及图 1 所示的两个外部数据存储—用户队列和消息队列。在实体图中,实体间的连线代表控制从一个实体流向另一个。如果没有箭头,表示相互作用的确切内容还没有决定。最高级的实体标为 EMS,它代表整个系统,是用户控制的实体。

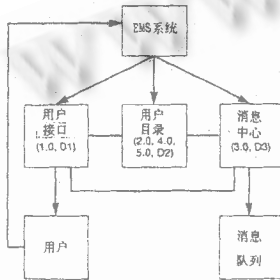


图 3 电子邮件系统的实体图

(5)定义对象。在面向过程和面向对象相结合的方法中,第五步是定义对象,是通过改变实体图来完成的,要对设计的简单性和对象抽象进行平衡,并搞清楚控制层次。例如,将用户目录与消息中心实体结合为一个对象简化了整体设计,但这个结合而成的对象不是一个紧凑的抽象观念。由于这个原因,在图 4 所示的对象图中还没有把它们结合起来。另一方面,电子邮件系统和用户界面实体在对象图中已结合成用户界面对象了。一旦用户启动系统,用户对象就作用系统的控制中心。

(6)定义变量和操作。第六步是为系统中的每个对象定义变量和操作(即,程序模块)。因为实体是由分组过程和数据存储定义的,所以变量和操作就通过巩固和推敲这些分组来定义。图 5 包括了一张可能的对象及与之相关的变量和操作列表。图 2 的“更改目录”过程在图 5 被分成了两个操作:“增加用户”和“删除用户”。除了那些和消息中心对象联系的变量和操作之外,变量一般

是对图 2 中数据调用的直接解释。这里将“用户索引”直接从图 2 中取出来,另外保留了四个变量(即正文、发送者 ID、接收者 ID、接收时间)来描述消息队列中的一个条目,这个队列是消息中心对象必须管理的。

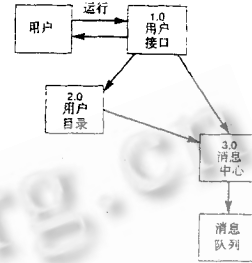


图 4 电子邮件系统的对象图

对象	变量	操作
用户接口	当前用户 ID 用户结点	确定用户显示菜单 增加用户 删除用户 列出用户 发送消息 读取消息 编辑消息
用户目录	用户 ID 用户名字 口令	注册 增加用户 删除用户 列出用户
消息中心	用户索引 正文 发送者 ID 接收者 ID 接收时间	接收存贮 检索发送

图 5 结合方法的电子邮件系统对象

(7)转换翻译。第七步是把对象同它的变量和操作(即方法)转换成面向对象的程序设计语言,这种语言应能提供对象类、变量和方法的定义。这一步是开发过程的结束,也标志着编码过程的开始。如果使用一种真正的面向对象的程序设计语言的话,编码是很容易的。

3.面向对象的系统设计

纯粹的面向对象系统设计方法也分为几步进行,在下面几段进行阐述。

(1)定义企业意图。纯粹的面向对象的方法是从依据商业企业的操作来定义其意图开始的。本文用作例子

的电子邮件系统可定义为一个使人们能发送和接收消息的系统。对于一个更加复杂的商业企业而言,需要用—个功能模型来定义它的意图。开发这样的模型是单独的一个课题,本文将在后面讨论。

(2)表示企业信息的特性。实现商业企业目标所需的信息必须用实体类型及其之间的关系来表示系统特性。在电子邮件系统中,其目标声明清楚地指出了它所需的信息是关于用户和消息的。对于一个更为复杂的企业,用于导出实体的信息必须由专家来决定。

在电子邮件系统中,用户和消息实体之间有两个关系是很重要的,即发送关系,定义了消息的发送者;接收关系,定义了每个用户的消息队列。在图6的实体关系图中描述了这些实体类型和关系。连接线末尾的几条分支线指明实体之间的关系可能是一对多和多对多的。

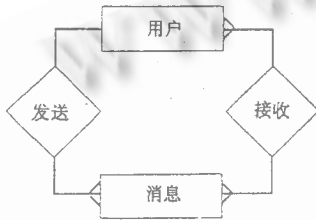


图6 电子邮件系统的实体关系图

(3)定义对象。单纯面向对象方法的第三步是依据系统实体的变量和操作来定义系统对象。电子邮件系统包括了三个对象的定义:命名了的用户、消息和接收(见图7)。从用户和消息实体类型直接引出了两个对象。第三个对象是接收对象,它是通过模拟每个用户的消息队列来实现接收关系。这一步完成了除屏幕设计之外的系统定义,提供了开始面向对象编码的先决条件。

4.两种方法的比较

面向过程和面向对象相结合的方法,与单纯的面向对象方法在设计过程方面有所不同,产生的系统设计结果也不同。对于电子邮件系统的设计来说,它们产生的三个对象都不一样。面向对象方法中的用户对象是结合方法中的用户接口和用户目录相组合而成的。结合方法无法将消息队列也组合进来,只把它当作系统外部的实体来处理。

面向过程和面向对象相结合的方法有三个不足之处,已通过电子邮件系统的例子反映了出来,下面几段将分别讨论。

对象	变量	操作
用户接口	用户 ID 当前结点 口令 用户名 发送的消息指针 接收的消息指针	注册 显示菜单 增加用户 删除用户 列出用户 发送消息 读取消息 编辑消息
消息	正文 下一个发送的指针 发送者指针 第一个接收者指针 接收时间	接收存储 检索发送 与发送者连接 与接收者连接 删除
接收	用户 ID 消息 ID 下一个接收的指针 下一个发送者指针	存储下一个接收的消息 存储下一个发送的消息 检索下一个接收的消息 检索下一个发送的消息 删除下一个接收的消息 删除下一个发送的消息

图7 纯粹面向对象方法的电子邮件系统对象

(1)复杂的系统设计过程。结合方法用六步才能完成的工作单纯面向对象方法三步就完成了。而且这六步比那三步要复杂。例如,结合方法的第二步,生成数据流图,就相当复杂,而且在设计过程的早期就已产生了详细的系统描述。在以后几步都要用到这个详细的数据流图,但使用不完全。

(2)松散地定义了对象。结合方法定义的对象缺乏紧凑性,因为这些对象反映了定义过程中的面向过程的观点。用户接口对象仅是一个过程集合,而没有作为对象的实体。用户目录和消息中心对象也是这样,它们实际上是过程分组而不是对象。

(3)数据的冗余存储。把用户界面接口和用户目录定义为单独的对象会造成了某些数据的冗余存储。系统还为每一个接收信息的用户分别存储了一份消息。与之相比,面向对象的系统只存储了一份消息清单,而使用指针将它们与接收者连接起来。还有一个更重要的冗余根源,这在电子邮件系统中没有表现出来,在由多个应用组成的系统中就会出现这种情况,即在这些系统中,每个应用系统都要有一份清单。

(下期续)