UML 模型的协调性检查

杨 静[1] 张明义[2]

(贵州大学计算机科学与工程学院 贵阳 550025)[1] (贵州科学院 贵阳 550002)[2]

摘 要 本文融合 UML 用例图、类图、顺序图和状态图,得到一个软件系统的需求模型,给出了这个需求模型的各个元素及相互间协调性检查的一种方法,这样,可以从软件开发的需求分析阶段检查模型的协调性,减少开发成本,最后再从协调的需求模型生成代码。

关键词 UML,用例图,类图,顺序图,状态图,需求模型,协调性

Consisitency Checking for UML Models

YANG Jing[1] ZHANG Ming-Yi[2]

(Conputer Science and Engineering College, Guizhou University, Guiyang 550025)[1] (Guizhou Science Academy, Guiyang 550002)[2]

Abstract A requirement model is obtained though integrating use-case diagram, class diagram, sequence diagram and state maching in UML, and consistency checking for the requirement model is proposed in the paper. Therefore, code can be generated from a consistent requirement model.

Keywords UML, Use-case diagram, Class diagram, Sequence diagram, State machine, Requirement model, Consistency

1 引言

在基于 UML 的软件开发过程中,几种 UML 的模型可以用来描述和分析系统开发过程中每个阶段所得到的产品。在 UML 的这种多视野的观点下,软件开发者可以把一个系统设计分解成较小规模可控模块。因此,软件开发者必须保证各种模型相互间在语法和语义上都是相容的。

近几年来,UML 模型的协调性检查各形式化分析是研究 UML 的热点之一[2~6,8],其中绝大部分工作主要是形式化 UML 的单个图(比如状态图)并仅仅处理一种图或二种图间的协调性。

本文旨在构建一个基于 UML 的软件开发的需求模型,分析这个需求模型的协调性,从而构建一个形式良好的需求模型。这个模型涉及到 UML 的用例图,类图,顺序图,状态图,可用于模型融合及代码生成。

2 UML 的用例图、类图、顺序图及状态图

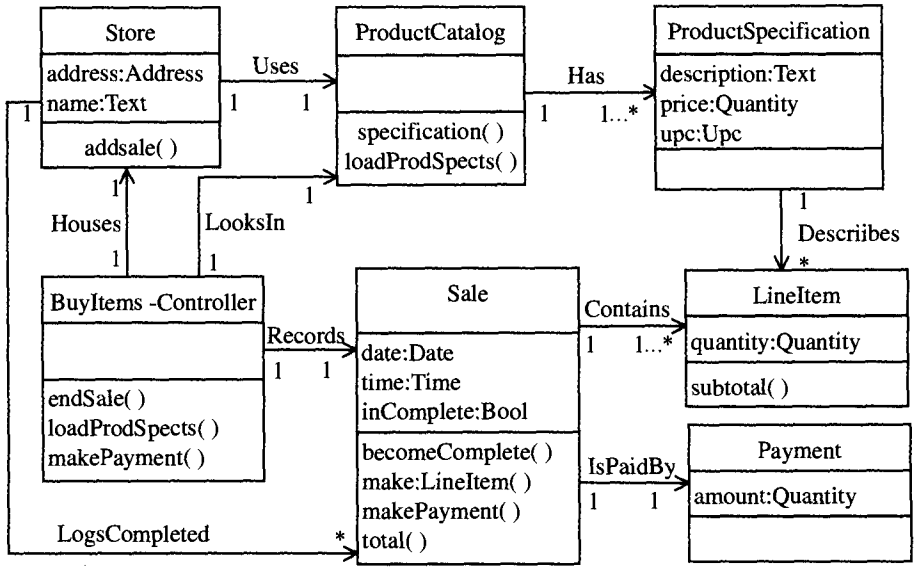

图 1 POST 系统的类图

在 UML 中,一个类图(Class Diagram)提供以下信息:

1)提供类上的静态信息和它们之间的继承关系:

- CN:由所有类所构成的有限集。
- $SUPER$:在集合 CN 上定义的直接的继承关系。

*)国家自然科学基金项目(No. 10161005),贵州省科研基金项目(No. 3086)。**杨 静** 博士研究生,副教授,研究方向:软件工程与形式化方法;**张明义** 教授,博士生导师,研究方向:计算机科学与人工智能。

2)描述类的结构。对集合 CN 中的一个类 C,用集合 $attr(c)$:

$\{\langle a_1,T_1\rangle,\cdots,\langle a_m,T_m\rangle\}$

表示 C 的基本属性,这里 T_i 代表类 C 的属性 a_i 的类型。

3)给出类之间联系的信息,用 $(C_1,m_1,A_\omega,m_2,C_2)$ 代表类 C_1 和 C_2 的有向联系 A_ω,用 $(C_1,m_1,A_\omega,m_2,C_2)$ 代表类 C_1 和 C_2 的无向联系 A_ω。这里 m_1 和 m_2 仅分别指多重性,如 1|0..1|*|1..*等等。

4)对集合 CN 中的每一个类 C,表示类 C 的所有方法的集合为 $method(C)$。

一个类图是形式良好的,其主要条件是继承关系不允许引进圈,另一个方面是类的命名问题。这里我们不考虑多重继承。图 1 是我们考虑的商店的自动收银(Point-of-Sale Terminal)POST 系统的类图。

一个类图定义了系统的状态空间,并且每个系统包含一些对象和对象之间的联系。因此,类图类似于程序中类的申明,类型及变量。一个系统状态其实就是这些变量良好的定义的状态。UML 完全通过用例机制来处理需求,但没有用例规格说明的任何形式化的结构。在我们的框架下,每一个用例 U 被模拟成一个用户类型控制类 U-controller(见图 2)。

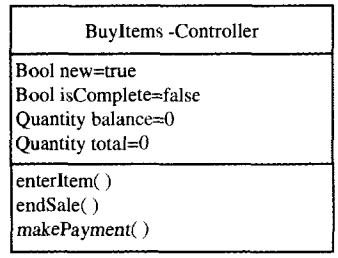

图 2 用例控制类 BuyItems

顺序图(Sequence Diagram)是用来描述为完成某项任务对象间相互通讯的模式。

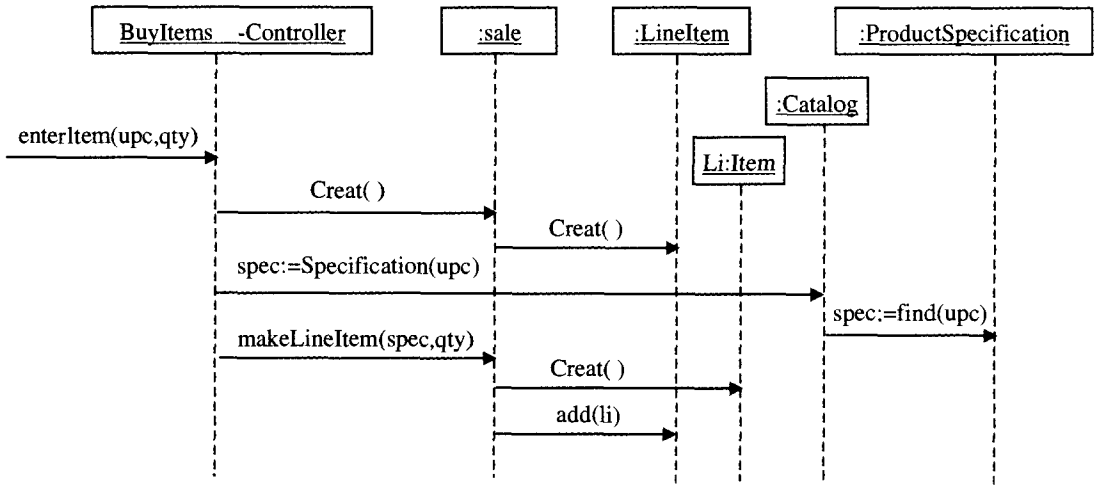

图 3 enterItem( )的顺序图

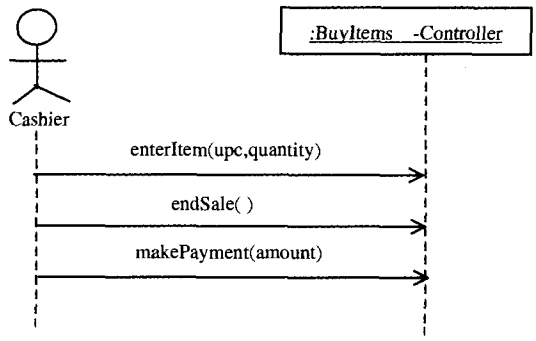

图 4 用例控制类 BuyItems 的顺序图

顺序图(如图 3 和图 4)的基本元素是信息,一条信息具有这种形式$\langle Ns,Action,Mt,ord\rangle$,其中:

- Ns 是一个类型为 N 的对象 s,它是信息发送者。
- Mt 是信息的接收者,其一个类型为 M,标识为 t。
- $Action$ 有两种形式:$g\rightarrow t.m$ 和 $*g\rightarrow t.m$,g 是作用于信息发送者 NS 的属性及公共变量上的布尔表达式,m 是类 M 中的一个方法,$*$ 是循环符号。
- ord 是偏序集$\langle\Lambda,\leqslant\rangle$的一个元素。其中集合 Λ 中的元素递归定义为

$n::=n|n.n,n\in NAT^+$,

$NAT^+=\{n\in NAT|N>0\}$

偏序$\leqslant$是标准的字典序。

顺序图就是一个满足下述条件的三元组$\langle Mo,m,MSG\rangle$:

- Mo 是调用方法 m 的接收者;
- m 是类 M 中的一种方法;
- MSG 是一些信息构成的一个有限集。

对每一个类,都用一个状态图(State Machine)来刻画这个类的对象的行为模式及其方法的功能,它包括对象的状态和某些事件(如调用方法和接收信号)的传送,我们只考虑简单的状态图,也就是没有复合态的状态图(见图 5)。一个状态图是一个四元组$\langle S,L,So,T\rangle$其中:

- S 是一个同状态构成的有限集;
- $so\in S$ 是初态;
- L 是同形如$\langle Event,Action,Guard\rangle$构成的有限集,称为符号集。对标签$\langle Event,Action,Gurard\rangle$而言,$Event$ 是该类的一个方法;$Action$ 是一些命令,$Guard$ 是这个类的属性及 $Action$ 上的局部变量上的布尔表达式,称为警卫;
- T 是一个关系:$T\subset S\times L\times S$,通常 T 中的每一个元素为一个切换。一个切换$\langle s,t,s'\rangle$把状态从 s 变到 s'。此时事件 $Event$ 发生了并且在状态 s 下 $Guard$ 成立,然后执行秩序 $Action$。

3 需求模型的协调性检查

正如文[7]指出的那样，一个软件系统的开发周期总是从需求模型的构建开始，在 UML 中，需求可由概念图和用例图来描述。概念类图是每一个类都没有方法的类图，而且类之间有的联系是没有方向的。因此，需求模式应包含一些 UML 模型：概念类图 Γ_r，一个用例图 U_r，一簇用例顺序图 Δ_r（每一个用例对应其用例顺序图，如图 4），一簇状态图 Ω_r（每一个用例有自己的一个状态图，如图 5）。如果考虑并发行为，那么一个需求模型也应包含一些活动图，本文中，不考虑并发行为。一个需求模型可用一个四元组 $\langle \Gamma_r, U_r, \Delta_r, \Omega_r \rangle$ 来表示，正如前面提到过的一样，每一个用例都可被模拟成一个用例控制类，其余类的所有属性和它们间的联系对用例控制类而言都有是可见的。用例图也提供了用例控制类之间的联系信息。如果用例 U_1 包含用例 U_2，那么从 U_1-*controller* 到 U_2-*controller* 有一个联系。用例图 Δ_r 描述的是参与者和系统间的交互。用例图已经被融合到类图中。

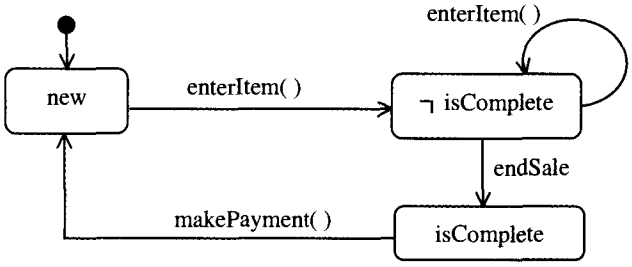

图 5 用例控制类 BuyItems 的状态图

跟随文[7]中提出的控制模式，对用例顺序图中出现的操作，我们在相应的用例控制类中做出相应方法的申明，并且状态图 Ω_r 应只与用例控制类有关。用例顺序图和状态图用来描述信息传送顺序和用例控制类中方法的行为。所以，对每一个用例 U，用例控制类 U-controller 申明了在用例顺序图中出现的操作，并且类 U-controller 中每个方法的方法体可用类 U-controller 的状态图中唤醒方法后的行动而得到。

从上述观点，可以概括出一个需求模型，如果满足下述条件是协调的：

1）每一个用户类型 U 的顺序图中出现的信息 m 被申明成 U-controller 中的一种方法。

2）每一个用例控制类 U-controller 的状态图中的事件 E-vent 是相应的顺序图中的一条信息的方法。

3）状态图中事件发生的秩序与顺序图中信息传送的秩序相同。

4）类的申明部份是良好的。

上述条件保证了每一个在用例控制类中使用的变量或者是基本型的，或者是类图中给出的一个类，在方法体中只能使用的相应类的属性，需求模型协调性保证了从概念图实现用例图的足够信息。

4 如何检查顺序图和状态图的协调性

需求模型的协调需要保证顺序图和状态图是协调的，而顺序图和状态图的协调要求状态图中事件发生的顺序和顺序图中信息传送的秩序相同，下面通过一个例子详细说明如何检查顺序图和状态图的协调性。给定一个顺序图，可以检查顺序图中的每一个对象的行为和这个对象所属类的状态图是否协调，采用一个例子[3]来说明我们的方法。这个例子是用来描述蜂窝电话与电话使用者之间的交互。在蜂窝电话系统中，一个用例是拨号。在这种用例模型下，我们考虑下列场景的发生：

1）用户正在输入一个电话号码；

2）此时有一个电话打入；

3）用户按下发送键。

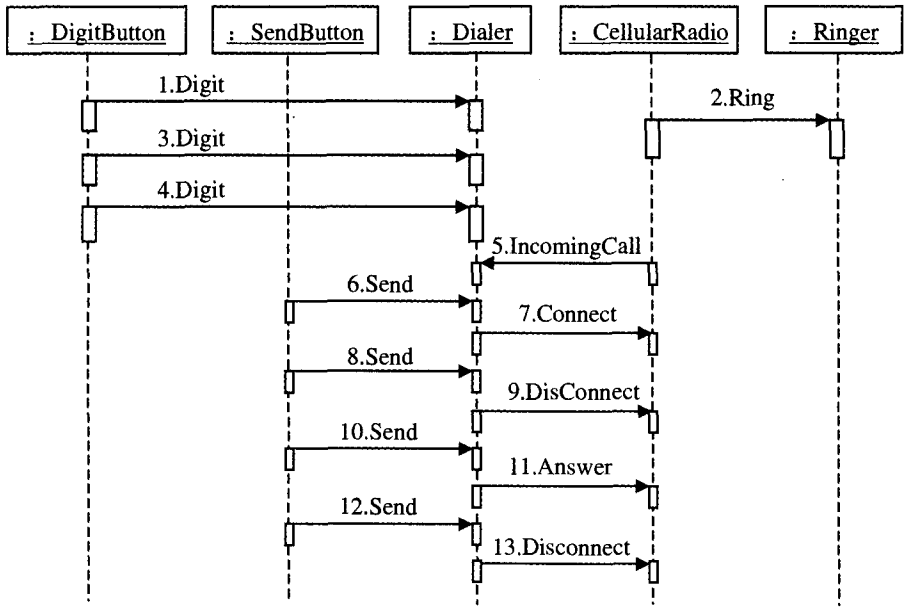

图 6 顺序图

正如我们所知，当用户按下发送键时意味着拨号类 Dialer 的对象或者在呼叫一个电话或者在接听一下电话。在上述场景下什么行为会发生？是接听电话还是呼叫电话？

顺序图和 Dialer 的状态图，分别在图 6 与图 7 中显示，这里我们省略了警卫和方法体。顺序图的设计者想呼叫电话，即使是按下发送键的情况下。而按照类 Dialter 的状态图，用户应按听电话。因此在顺序图和 Dilar 的状态图之间发生了不协调问题。下面将描述如何检查顺序图和状态图的协调

性，其方法是从顺序图按照时序产生一串标签对并附在状态图上，对比这组标签和状态图本身的标签，按照下面的算法产生结论。给一个类 C 的对象 O，把所有发送者是 O 或接收者是 O 的信息搜集在一起，并按在顺序图中出现的时间顺序排列成：

$$msg_1, msg_2, \cdots, msg_n \quad (1)$$

其中 msg_i 是离 msg_{i+1} 最近先发出的一条信息，注意可能有 n 条类似式(1)的链，这时我们需检查每一条链。一条信息可以改变状态图当前对象的状态。顺序图中的一条信息可以大致反映了对应状态图的以下两种情况：

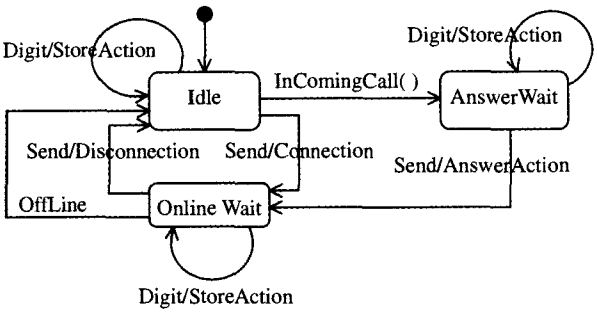

图 7　类 Dialer 的状态图

1)如果这条信息的接收者是给定对象，那么这条信息对应于触发事件 Event；2)如果信息的发送者是给定对象，那么这条信息对应于一个动作 Action；

形式化地，可以从链(1)中通过下面的方法得到标签：对于链中的接收对象是被检查对象的信息，我们获取标签〈Event，Acion，Guard〉，其中 Event 必须是相应信息中方法的签名，Guard 是这条信息的警卫，如果下列一条信息的接收者是正在被检查的对象，Action 就是这条信息的方法体。否则，如果下一条信息的发送者是正在被检查的对象，Action 是后一条信息中方法体。这样一来，我们可以从这个链中产生相同时序的一组标签。现在通过交替遍历这簇标签各状态图，我们可以比较这簇标签和状态图，可以比较这簇中出现的标签与状态图是否相同，从而断定相应的顺序图和状态图是否协调，其算法如下：

step1：对第一个标签，我们从初态 s_0 遍历状态图，如果状态图中从 s_0 出发的所有切换中正好有一个标签与此标签相同，那么得到后继状态 s'，并把这个标签附在状态 s' 上，转 Step-2；如果没有标签与此相同，转 Step-4；如果有两个以上的标签与此标签相同，转 Step-5；

Step-2 对第二个标签，从当前对象状态遍历状态图，当前对象状态也就是附着第二个标签的状态，如果状态图中从当前对象状态出发的所有切换中正好有一个标签与此标签相同，那么把下一个标签附在后继状态上，用后继标签替换第二个标签后反复执行 Step-2；如果没有标签与此相同，转 Step-4；如果有两个以上的标签与此标签相同，转 Step-5；

Step-3 如果我们能按照产生出来的标签的时序成功遍历状态图是协调的，退出；

Step-4 顺序图各状态图是不协调的，退出；

Step-5 状态图具有二义性，退出。

对于蜂窝电话系统，遍历 Dialer 的状态图和顺序图的信息 6 产生的标签，被附在状态 Answerwait 上，再从当前对象状态 Answerwait 上遍历 Dialer 的状态图，找不到相同标签的任何切换，由此可知，这个状态图和顺序图是不可协调的，这个系统设计存在错误。

Dialer Sequence Diagram			Dialer State Diagram		
Message	Generating Label	Attached State	Current Object State	Transition Label	Resulting Object State
1.Digit	Digit/StoreAction		Idle(Initial State)	Digit/StoreAction	Idle
3.Digit	Digit/StoreAction	Idle	Idle	Digit/StoreAction	Idle
4.Digit	Digit/StoreAction	Idle	Idle	Digit/StoreAction	Idle
5.IncomingCall	IncomingCall	Idle	Idle	IncomingCall	AnswerWait
6.Send **	Send/Connection	AnswerWait	AnswerWait	Send/AnswerAction	Error **
8.Send					
10.Send					
12.Send					

图 8　交替遍历顺序图和 *Dilaler* 状态图

结论　我们提供了一个融合的 UML 模型，从而构建了一个需求模型，这个模型符号协调性可通过一个算法作静态的检查，而不变量可用模式检测工具检测，基于这个需求模型，可用基本的程序设计技术从这个需求模型转化到代码生成。将来的工作包括融合活动图到我们的框架下分析并发行，并用此框架开发相应的需求模型的协调检测工具。

致谢：感谢贵州大学人工智能实验室的官云鸿和王以松对本文的大力支持和帮助。

参 考 文 献

1　方贵宾，等译、UML 和统一过程。机械工业出版设。2003

2　Back R，Mpkhajlova A，von Wright J. Class refinement as semantics of correct object substitutability. Formal Aspects of Computing，2000，2：18～40

3　Tyszberowicz S，Litvak B，Yehudai A. Behavioral consistency validaion of uml diagrams. In：list IEEE International Conference on Software Engineering and Formal Methods(SEFM)，IEEE Computer Society，2003. 118～125

4　Engels G，et. al A methodology for specifying and analyzing consistency of object-oriented behavioral moderls. In：The Proc. FSE-10，Ausia，2001

5　Kuester J M，Engels G，Groenewegen L. Consistent interaction of software components. In：IDPT2002，2002

6　Yang J，Long Q，Liu Z，et al. A predicative semantic model for intergrating uml models. In：Z. Liu and K. Araki，eds. LNCS3407 Berlin Heidellberg，2005

7　Larman C. Applying UML and Patterns. Prentice-Hall Iternational，2001

8　Reggio G，et al. Towards a rigorous semantics of UML supporting its mulriview approach. ，LNC2029. Speringer，2001