从用例模型到基于使用测试模型的转化

On the transformation of Use Case Model to Usage-based Test Model

凌 辉 李 茜 许晓春 徐永森 徐家福

(南京大学计算机科学与技术系 计算机软件新技术国家重点实验室 南京210093)

Abstract Requirements specification and system test are two important aspects in software development process. In this paper we focus on Use Case Model(UCM)in requirements specification and statistical usage test(SUT)in system test, and present a transformation approach from UCM to state hierarchy model, which is the underlying model of SUT. Both advantages and disadvantages of this approach are discussed.

Keywords Transformation, Use Case, UCM, SUT, State hierarchy model

1 引言

需求分析和系统测试是软件开发过程中非常重要的两个方面。系统测试包括功能测试和非功能测试、功能测试的主要目的是通过测试判断系统是否正确实现了功能需求，非功能测试中最主要的是可靠性测试即该系统的使用是否可靠。系统测试需要使用需求分析得到的相关信息，两者关系比较紧密。如何将两者集成起来是本文主要讨论的问题。Regnell[1]提出了将两者集成起来的两种方法：转换和扩充，并给出了从层次用例模型(Hierarchical Use Case Modelling)[2]到基于使用测试(Usage-based Testing)的转换过程。大家知道，目前最常用的用例模型(UCM)不是层次用例模型而是UML(Unified Modelling Language)[3]中的一种相关模型，因此构造从UML中的相关用例模型到基于使用测试模型的转化过程显然更为实用。用例模型是Jacobson[4]在1992年提出的一种面向对象软件开发方法中所使用的模型，主要用于系统需求分析阶段理解和描述需求信息。统计使用测试(SUT)是Mills等人[5]在1987年提出的一种基于系统使用统计的测试技术，该技术通过构造系统使用模型而得到一些测试实例。这些测试实例可以用于系统功能测试以及可靠性预测和验证。我们以用例模型为需求分析的模型，以统计使用测试为系统测试的模型，通过构造用例模型到统计使用测试的转化过程实现需求分析和系统测试的集成。

2 用例模型相关概念

为了简单明了地阐明从用例模型到统计使用测试的转化过程，我们以一个简化的用户交换机(pbx)为例展开全部讨论。pbx是通信过程中使用的用户交换机的简单模型，它提供三种电话服务：普通付费电话(NCC)、无条件拨号(CFU)和标志读取与复位(RMR)，如表1所示。其中电话用户使用NCC和CFU，而操作员使用RMR。

软件需求分析是软件开发过程中非常重要的一步，必须使用定义明确的概念和方法才能完整地、一致地、清晰地把需求信息表示出来。用例模型就是这样一种方法。自从Jacobson提出了用例模型之后，许多面向对象开发方法中都使用了用例技术，如：OMT[6]、the Booch method[7]、ROOM[8]。

下面介绍用例模型基本概念以及pbx的用例模型表示。

表1 pbx中的服务

服务	描述
NCC	普通付费电话
CFU	无条件拨号
RMR	标志读取及复位

2.1 定义

服务 是系统向用户提供的一组功能特性。

使用实体 表示系统外部同用例进行交互的实体，可以是某个人、其它软件系统、也可以是其它的用例。对于pbx，使用实体有使用者和操作员。

目标 是用户使用系统服务所要达到的目的。

根据目标把用户分成不同的使用实体，一个使用实体代表一类用户，他们出于同样的目标，以同样的方式使用系统，pbx中的使用实体及其目标如表2所示。

表2 pbx 中的使用实体及其目标

使用实体	目标
使用者	Gs1：同另一使用者建立通信
	Gs2：停止同另一使用者的通信
	Gs3：建立到另一位置的可到达性
	Gs4：停止到另一位置的可到达性
操作员	Go1：维持通信期间标志信息
	Go2：打印输出每个使用者的标志数
	Go3：使用者标志数复位

用例 是系统执行动作的一个序列，这组动作的执行将向与该系统交互的使用实体(Actor)返回可以度量的结果。pbx 中所有的用例如表3所示。

用例给出系统为完成某个功能而执行的一个具体的动作序列，即系统的一个事件流。通常至少有一个情景(Scenario)同一个用例相对应。情景是使用实体同系统之间的一个交互序列，这个序列的所有情况都是确定的。因为可能出现选择和重复，所以一个用例可能包含了无数个情景。而一个情景则是一个用例的具体、特定的实现。表4列出了用例 Normal Call 的所有情景。

表3 pbx 中的 Use Cases

Use Cases	使用实体	目标	服务
Normal Call	使用者	Gs1，Gs2，	NCC
Activate CFU	使用者	Gs3	CFU
Deactivate CFU	使用者	Gs4	CFU
CFU Call	使用者	Gs1，Gs2，Gs3，	CFU
Read Markings	操作员	Go1，Go2	RMR
Reset Markings	操作员	Go3	RMR

表4 Use Case"Normal Call"的情景

情景	描述
Reply	被叫使用者回答
Busy Subscriber	被叫使用者忙
No reply	被叫使用者不响应
Non-Existent	被叫使用者不存在
Timeout	使用者超时

这样一个应用系统可以表示成如下的三元组：

A=(ServerSet，ActorSet，UseCaseSet)

其中 A 是应用系统，ServerSet 是系统服务的集合，ActorSet 是使用实体集合，而 UseCaseSet 是系统用例的集合。

例如 pbx=({NCC，CFU，RMR}，{使用者，操作员}，{Normal Call，Activate CFU，Deactivate CFU，CFU Call，Read Markings，Reset Markings})。

2.2 用例表示

为了准确描述用例的相关信息，UML 中使用了状态图(State Diagram)、序列图(Sequence Diagram)和协作图(Collaboration Diagram)[9]。在从用例模型向测试模型转化过程中，主要使用了状态图和序列图，因此我们主要介绍这两种图，关于协作图的详细内容请参考文[9]。

状态图 描述了一个对象或交互中的对象在接受消息做出响应，以及执行动作的整个过程中所经历的状态序列。状态指的是对象活动期间满足某个条件或执行某个动作或等待某个事件的一种情况。对象在一个状态保持有穷时间。我们给出状态图的数学表示。

State=(name，state_variable，Internal_activity)；
Tran=(state1，state2，event)；
StateSet={State_1，State_2…State_n}；
TranSet={Tran_1，Tran_2…Tran_m}；
StateDiagram=(StateSet，TranSet)；

图1是用例"Normal Call"的状态图。

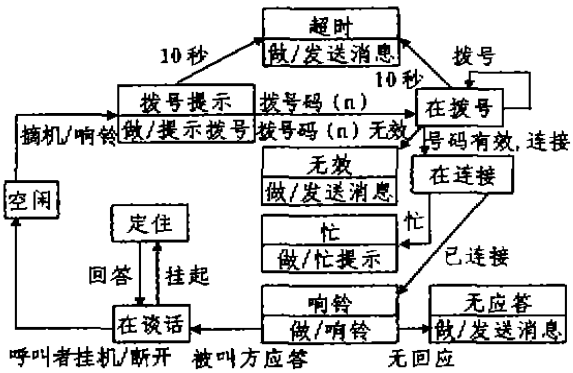


图1 用例 Normal Call 的状态图

序列图 以时间序列来表示对象行为，它描述了参与动作的所有对象以及消息交换序列。序列图可以描述用例所有的情景也可以只描述其中一个。

Interaction=(sender，receiver，message)
InteractionList=(Interaction_1，Interaction_2，…，Interaction_n)
SequenceDiagram=InteractionList；

如果 i<j，那么 Interaction_i 在 Interaction_j 之前发生。

图2描述了用例"Normal Call"的情景 Reply。

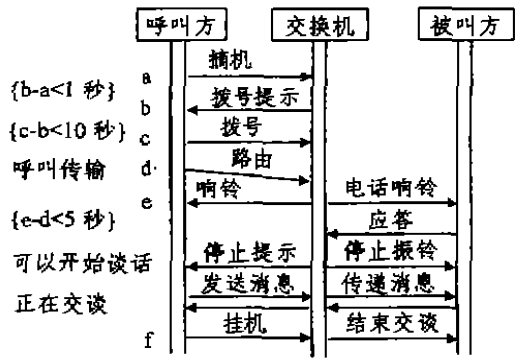


图2 Reply 情景的序列图

3 可靠性需求规约和验证

可靠性需求是除了功能需求之外需求分析中最重要的一部分，系统可靠性不仅和系统特性有关，而且同系统环境、系统使用相关。基于使用的测试在功能确认的同时进行可靠性验证。下面以 pbx 为例介绍统计使用测试模型。

3.1 基于使用测试概念

所有测试技术的主要目的都是确认系统实现了用户需求，特别是功能需求。目前大部分测试集中在功能需求测试上，但同时还应该重视其他质量因素，如尽可能多地定位错误，软件可靠性等。

基于使用的测试侧重于发现引起最常见失败的错误，从而尽可能提高系统可靠性。基于使用的测试使用面向用户的方法，需要对软件系统预期使用情况及预期使用的数量信息建模。在这方面已经研究了多种方法如 Musa 提出的操作文件测试[10]，Mills 等人在1987提出的基于操作文件的随机测试[5]以及 Runeson 和 Wohlin 在1995年提出的基于操作文件的用户状态相关随机测试方法[11]。本文使用的测试方法是基于状态层次模型[12,13]的统计使用测试方法。

图3描述了基于使用测试过程中软件系统和系统环境概念之间的关系，这些概念是创建软件使用模型及量化系统使用信息的基础。

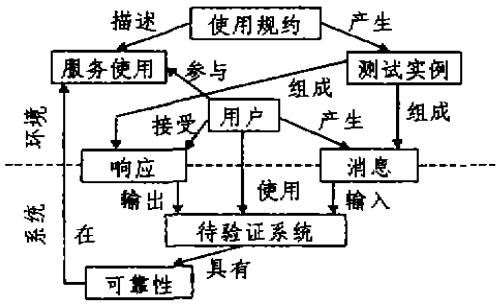


图3 基于使用测试概念及其关系

使用规约描述了系统环境相关信息，其中使用模型描述了用户所有可能行为；使用侧面（Usage profile）分成层次侧面（hierarchy profile）和行为侧面（behavior profile），层次侧面描述了选择某个用户的可能性，而行为侧面则说明了单个用户使用系统服务时的行为。图4描述了使用测试模型概念之间的关系。

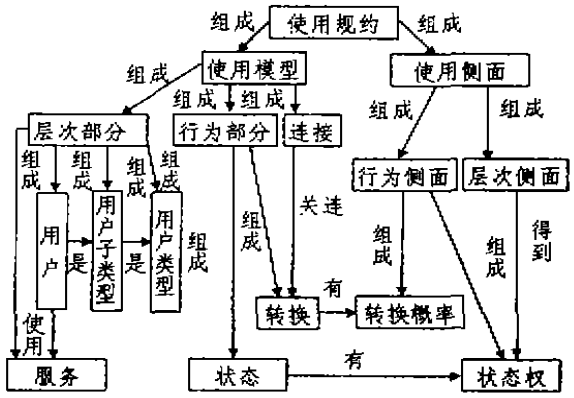


图4 使用测试模型概念

3.2 用户行为的层次表示及使用侧面

使用模型主要由三部分组成：层次部分、行为部分和连接。行为部分描述了系统服务的行为模型，而层次部分则将系统使用按不同的级别进行划分。连接定义了该行为模型与其它行为模型之间的关联。如图5所示。

对应每一个服务的实例，即服务被某个用户的一次使用，都有一个相应的行为模型，行为模型由状态和转换组成。这里的状态是指外部状态，即用户状态，是系统状态的一个子集。

对每个状态赋一个权值 W_k，W_k 表示用户在该状态下使用服务 k 的频率，也就是从该状态转移到服务 k 的行为模型中的状态的频率，对每条边也赋一个值 V_{ij} 表示用户在状态 i 时转移到状态 j 的可能性，每个状态到它所有相邻状态的 V 值的和为1。

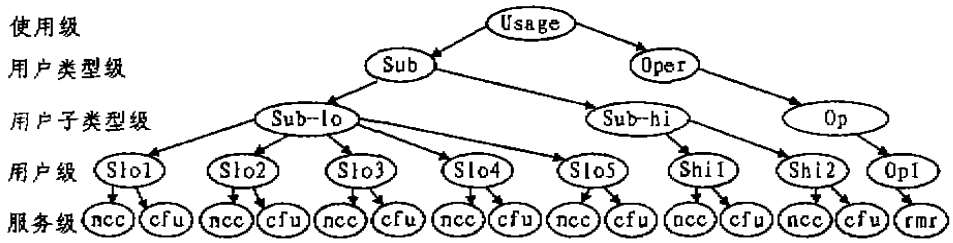


图5 pbx 使用模型的层次部分

在图5中待验证系统的使用分解成单个用户和其所使用的服务，同一类型的所有用户具有相同的行为模型，根据不同的行为侧面划分用户子类型，同一子用户类型的所有用户具有相同的行为侧面，pbx 中的用

户类型分成 Subscriber-Low 和 Subscriber-High 两个子类型。S-Lo1—S-Lo5 和 S-hi1—S-hi2 分别属于这两个子类型，它们具有不同的行为侧面，但具有相同的行为模型、使用相同的服务 NCC 和 CFU，操作员用户类型由一个用户子类型 Op 组成，对应 Op 子类型只有一个用户 Op1，Op1 使用服务 RMR。层次侧面则比较复杂，因为当用户状态发生改变时，层次侧面中用户使用某个服务的可能性(Pk)也要发生变化，我们用如下的公式计算 Pk，

$$Pk=\begin{cases} Wk/(\Sigma Wj), \text{当存在 } j \text{ 使 } Wj\neq 0 \text{时} \\ 0 \quad , \text{否则} \end{cases}$$

对应行为模型中的每个转换，层次侧面都需要更新，更新算法参考文[13]。

利用上面介绍的测试模型就可以产生相应的测试实例，对系统可靠性及功能需求进行测试。具体过程请参考文[14]。

4 从用例模型到基于使用测试的转化

Wohlin[11]指出：用例和基于使用测试的结合提供了一个从用户角度综合观察软件开发过程的方法，用户只需集中注意系统的外部视图和实际使用而无需关心系统内部实现。因此用例模型和基于使用测试的集成可以让用户在更抽象的层次上理解系统。虽然在用例模型中也使用了类似的概念和信息，但并不存在概念之间的一一映射，在从用例模型到使用模型的转化过程中必须消除这些概念之间的差异。由于在用例模型中没有系统使用信息，我们必须从其它途径收集这些信息，创建使用侧面。

从对用例模型和基于使用测试模型的分析可知，测试模型的构造主要是使用模型和使用侧面的构造。使用侧面构造在使用模型的基础上进行。因此首先要建立使用模型。在建立使用模型之前必需分析出用户类型、服务和转化等基本组成元素。我们根据使用系统服务的不同来区分用户，定义用户类型，按照服务构造行为模型，所以首先必须在用例模型的基础上识认出系统的所有服务，然后划分用户类型，在这基础上收集其它使用信息定义用户子类型，收集所有用户实例，构造使用模型层次部分。之后，我们从用例模型中提取信息为每个服务构造行为模型。在建立了使用模型之后，再根据使用信息定义行为使用侧面和层次使用侧面。测试模型的各个部分的构造有一定的先后顺序，因此从用例模型到基于使用测试的状态层次模型的转化也必须遵循一定的步骤和方法。

为了消除用例模型和使用模型之间的概念差异，实现从用例模型到状态层次模型的转化，我们采用如下方法构造状态层次模型：

·识认服务
·定义用户类型
·定义用户子类型及用户实例
·创建行为模型
·定义行为使用侧面
·定义层次使用侧面

上面这些步骤不是顺序的，必要时需要重复。

识认服务 我们定义一个过程：Identify_S(UCMSer,SUTSer)进行服务识认，该过程在用例模型的服务中提取，得到统计使用测试模型中使用的服务。

pbx 直接使用用例模型中定义的服务。最后得到的服务就是 NCC、CFU 和 RMR。

定义用户类型和子用户类型 我们定义两个过程：

Define_UserType(UCMActors,SUTUserType)
Define_SubUser(SUTUserType,OtherInfo,SubUserType)。

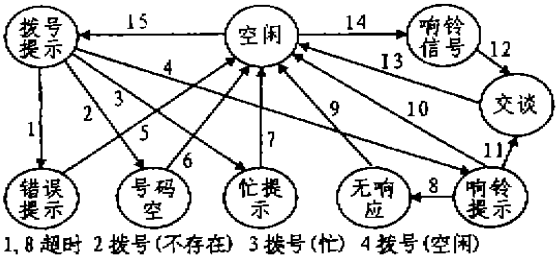
Define_UserType 对用例模型的使用实体进行分析得到 SUT 中的用户类型。用例模型中的使用实体是层次模型中用户类型的基础。统计使用测试是从系统使用角度进行测试，它的用户类型指的是系统的外部用户，而用例模型中的使用实体不仅包括外部用户，还有可能是其它的用例以及系统，所以需要分析使用实体，提取出系统的外部用户。Define_SubUserType 过程对用户类型进行分析得到用户子类型。使用 Define_UserType 过程从用例模型得到用户类型后，再从其它地方收集用户使用服务的数据信息 OtherInfo，根据这些信息划分用户子类型。接下来以 pbx 为例具体阐述这一过程。

pbx 中有两类使用实体：使用者和操作员，这两类使用实体都是外部用户，所以相应的就有两个用户类型。Subscriber 用户类型使用 NCC 和 CFU 服务，而 Operator 用户类型使用 RMR 服务。Subscriber 用户类型又可以分为两类：一类经常使用服务，一类不常使用。这样就得到两个用户子类型 Subscriber-Low 和 Subscriber-High。层次模型分析结果如图5所示。

创建行为模型 创建过程分成两步：Define_State(StateDiagram,StateSet)和 Define_Transition(StateDiagram,SequenceDiagram,TransitionSet)。在用例模型中没有直接可用于创建行为模型的相关信息，但是可以从用例状态图表示和序列图表示中提取相关信息。行为模型是一个状态转换图，它可能包含整个用例也可能只包含一个或几个情景。Define_State 分析用例状态图的状态集合，提取出与用户类型相关的状态，得到行为模型的状态集合。Define_Transition 分析用

例状态图和序列图，提取可能引起用户状态变化的消息，得到行为模型的转换集合。根据状态集合和转换集合，就能得出行为模型的图形表示。

我们以用例 NCC 的情景 Reply 为例具体阐述。如图6所示，我们首先定义用户起始状态"空闲"，这时 Subscriber 用户唯一能发出的消息是摘机，导致用户状态从空闲转化成拨号提示，接下来输入号码，用户状态变成响铃提示，被叫方回答，用户状态变为交谈，最后谈话结束，用户发出挂机消息，回到空闲状态。这样我们就得到了 Reply 情景的行为模型，把其它情景考虑在内，就得到了服务 NCC 的行为模型，如图6所示。其中带 * 的转换如 B—Answer 和 B—Calling 表示包含了另外一个行为模型，该行为模型的状态转换会引起当前模型的一个状态转换。



1, 8 超时 2 拨号(不存在) 3 拨号(忙) 4 拨号(空闲)
5, 6, 7, 9, 10, 13 挂机 11B-Answer* 12, 15 摘机 14B-Calling*

图6 服务 NCC 的行为模型

定义行为使用侧面 创建完行为模型之后就可以开始定义行为使用侧面，由于在用例模型中没有系统使用信息，必须从其它地方获取相关信息。

表5 "Subscribe"用户类型行为侧面

状态	转化	Subscriber -Low 类型	Subscriber -High 类型
空闲	摘机	1.0	1.0
拨号提示	拨号(空闲)	0.70	0.60
	拨号(忙)	0.25	0.35
	拨号(不存在)	0.03	0.03
	超时	0.02	0.02
响铃信号	挂机	1.0	1.0
响铃提示	挂机	0.98	0.98
	超时	0.02	0.02
交谈	挂机	1.0	1.0
忙提示	挂机	1.0	1.0
无响应提示	挂机	1.0	1.0
错误提示	挂机	1.0	1.0
号码空	挂机	1.0	1.0

表5描述了对应于 NCC 服务行为模型的用户于类型 Subscriber-Low 和 Subscriber-High 的行为使用侧面。表6则描述了对应于 NCC 服务行为模型的用户于类型 Subscriber-Low 和 Subscriber-High 的状态权值。空闲状态的权值表明 Subscriber-High 开始打电话的可能性比 Subscriber-Low 要高一倍，交谈状态的权值表明 Subscriber-High 交谈时间要比 Subscriber-Low 长一半。

定义层次使用侧面 最后，根据每个服务的状态权值，按照如下公式就能计算出层次使用侧面内容。

$$p_i = W_i / \Sigma W_k \quad (k = 1 \cdots 15)$$

表6 状态权值表

状态	Subscriber-Low 类型状态权值	Subscriber-High 类型状态权值
空闲	1	2
拨号提示	100	100
响铃提示	100	100
响铃信号	50	50
交谈	15	10
忙提示	100	100
无响应提示	100	100
错误提示	100	100
号码空	100	100

结束语 从上面的分析可以得出，只要建立合适的转换方法及添加其它必要信息，从用例模型到基于使用测试模型的转化是可行的。这种转化方法的优点是：它建立在两个比较成熟的模型之上，只要从用例模型构造了相应的测试模型，就可以使用现有的测试技术实现对用例的测试，缺点是需要进行模型间的转化。

本文介绍了用例模型和基于使用测试模型的基本概念，分析了这些概念之间的关系，通过建立转化方法实现从用例模型到基于使用测试模型的转化。显然这个转化简单可行，今后将进一步给出转化方法的正确性证明。

参 考 文 献

1 Regnell B, et al. Towards Integration of Use Case modelling and usage-based testing. The Journal of System and Software, 2000, 50: 117～130

2 Regnell B. Hierarchical use case modelling for requirements engineering: [Technical Report 120]. Department of Communication Systems, Lund University, Tech. Lic. Dissertation. 1996

3 The Object Management Group. UML Summary, Version 1.0.1(1997-3-19)

4 Jacobson I, et al. Object-Oriented Software Engineering—A Use Case Driven Approach. Addison-Wesley Press, 1992

5 Mills HD, Dyer M, Linger R C. Cleanroom software engineering. IEEE Software, 1987(Sep.): 19～24

(下转第51页)

定必然正确的结论，这和人类用常识处理非全知世界所采用的方法很相近。而且，常识推理的研究已经找出了超协调逻辑和非单调逻辑的统一新形式，如超协调非单调逻辑[15]。

最后，在理解了以上几个观点之后，容易想到这类形式方法最大的难点在于使不精确的需求假设空间更加合理。用背景知识和领域知识约束推理的范围可以提供为人接受的、符合常识的形式规格，解决由于有限资源不能满足很多需求时优先级的确定问题，避免诸如彩票悖论之类的矛盾，非事实知识[16]就是在部分领悟人类认识认知和智能行为的基础之上，用隐含于事实之后的环境知识限制讨论范围，大大提高了软件系统的健壮性。然而，如何获取非事实知识则不是一件容易的事，牵扯到知识发现、学习和归纳等很多领域。使用各种各样的方式嵌入推理系统的假设在受限（包括显式和隐式）软件系统和应用之间架起了桥梁。因而，研究这个问题是非常必要的，也是很有现实意义的。

限于篇幅，这里只给出了大致解决问题的几点考虑，我们将在后续文章中分别和逐步地给出我们自己在这几方面的工作进展。

结论与展望 本文把知识工程的一些思想和方法引入到需求工程领域，指出特定背景下需求合适的表示方法；用常识知识帮助限制推理范围、确定优先级和认识智能行为；提出了应以广义模糊逻辑理论为基础，深入研究需求工程中的不确定性问题，推广需求规格形式化的适用范围。

参 考 文 献

1 唐稚松. 时序逻辑程序设计与软件工程. 科学出版社，1999

2 Krasner H. Requirements Dynamics in Large Software Projects, A Perspective on New Directions in the Software Engineering Process. In: Proc. IFIP. Elsevier, New York. 211～216

3 Jarke M, Pohl K. Requirements Engineering in 2001: (Virtually) Managing a Changing Reality. Software Engineering, 1994(Nov.): 257～266

4 Potts C, Takahashi K, Anton A. Inquiry-based Requirements Analysis. IEEE Software, 1994(Mar.): 21～32

5 Feather M, Fickas S, Kramer J. Living with Inconsistency. In: Proc. ICSE'97 Wordshop. Boston, May 1997

6 Finkelstein A, et al. Inconsistent Handling in Multi-perspective Specifications. IEEE Trans. on Software Engineering, 1994, 20: 569～578

7 Boehm B, In H. Identifying Quality Requirement Conflicts. IEEE Software, 1996(March): 25～35

8 Elkan C. The Paradoxical Success of Fuzzy Logic. In: Proc. 11th National Conf. on Artificial Intelligence. MIT Press, 1993. 698～703

9 Batyrshin I, Kaynak O. Parametric Classes of Generalized Conjunction and Disjunction Operarins for Fuzzy Modeling. IEEE Transactions on Fuzzy System, 1999, 7(5): 586～595

10 何华灿，刘永怀，等. 经验性思维中的泛逻辑. 中国科学(E辑)，1996, 26(1): 72～78

11 何华灿，刘永怀，等. 命题泛逻辑学原理. 科学出版社(待出版)

12 Williams M-A, Antoniou G. A Strategy for Revising Default Theory Extensions. In: Proc. 6th Intl. Conf. on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, 1998. 24～33

13 Yager R R. Fuzzy Sets Methods for Representing Commonsense Knowledge. Journal Intelligent and Fuzzy Systems, 1999, 7: 27～45

14 Rocha M. Evidence Sets: modeling subjective categories. International Journal of General System, 1997, 27: 457～494

15 林作铨. 超协调限制逻辑. 计算机学报，1995, 18(9): 665～670

16 Hoffmann A. Non-Factual Knowledge. Paradigms of Artificial Intelligence, Springer, 1998. 243～252

（上接第34页）

6 Rumbaugh J, et al. Object Oriented Modelling and Design. Prentice Hall, New Jersey, 1991

7 Booch G. Object Oriented Design with Applications. Benjamin/Cummings, Menlo Park CA, 1994

8 Selic B, et al. Real-Time Object-Oriented Modelling. John Wiley & Sons, Inc. New York, 1995

9 The Object Management Group. UML Notation Guide, Version 1.0(1997-1-13)

10 Musa JD. Operational profiles in sofrware reliability engineering. IEEE Software, 1993(March)

11 Runeson P, Wohlin C. Usage modelling: the basis for statistical quality control. In: Proc. of the 10th Annual Software Reliability Symposium. Denver, Colorado

12 Wholin C, Runeson P. Certifications of Software Components. IEEE Transactions on Software Engineering, 1994, 20(10)

13 Runeson P, Wohlin C. A dynamic usage modelling approach to software reliability engineering: [Thesis]. Department of Communication Systems, Lund University, Lund Sweden

14 Wesslen A, Wohlin C. Modelling and generation of software usage. In: Proc. Fifth Intl Conf. on Software Quality