基于 UML 和 Petri 网的用户界面原型的研究

Research on User Interface Prototyping Based on UML Scenarios and Petri Nets

魏定国[1,2]　吴时霖[1]

(复旦大学计算机系　上海200433)[1](广东商学院　广州510320)[2]

Abstract　In this paper we discuss a requirement engineering process how to generate a user interface prototype from scenarios and yield specification of system in form of Colored Petri net. Thereafter scenarios are described in form of sequence diagrams defined by the Unified Modeling Language (UML) with additional user interface information. These diagrams are transformed into Petri net specifications which can capture the behavior of the entire system. From global specification, a user interface prototype is generated and embedded in a user interface builder environment for further refinement. With end user's feedback and verification, the user interface prototype may be iteratively refined.

Keywords　User interface (UI), Prototyping, Scenario specification, Colored

1　引言

众所周知,概要是理解需求和分析人机交互最有效的方法。一个典型的基于概要的需求工程有两个主要的任务。一是生成描述系统行为的概要规格说明;二是用户通过仿真和原型开发来验证概要的有效性。如果没有支持这项工作的自动工具,这仍然是一件费时乏味的工作。

在开发早期的阶段,快速原型开发非常有效,因而得到了普及和广泛的使用。最近,虽然用户界面原型开发工具都有了很大的改进(如 UI builders 和 UI 管理系统),但用户界面的开发仍然是一个费时的工作,因为必须建立每一个 UI 对象,并准确地安置它们,而且每一个对话规格说明还要通过编程(如 UIbuilders)或专门的语言(如 UI 管理系统)加上去。

本文为需求工程提供一种基于统一建模语言(UML)和着色 Petri 网的用户界面开发方法。它是一个交互式过程,分四步进行,只需要有限的人工干预就可以从概要导出用户界面的原型,并生成系统的正式的规格说明。在此过程的第一步,用 UML 中所定义的用例图详细描述系统,对系统的用例图中发生的每一个用例,其概要应以 UML 序列图的形式给出,并附上丰富的 UI 信息。第二步将用例图和所有的序列图转换成着色 Petri 网(CPNs)。第三步,将各个 CPNs 描述的用例图合并集成单个的 CPN,将所有以这种方式从用例图得到的 CPNs 联接在一起,形成全局的 CPN 来捕捉整个系统的行为。最后,系统的 UI 原型可以从全局的 CPN 中产生,并能嵌入到用户界面开发环境中以便进一步求精。

在此方法中,其目的是要在用例和概要的层次上分别建模,并希望在它们集成之后还能跟踪概要。因此我们需要支持层次、着色的 PN 类或对象来区分所得出的规格说明的概要。我们采用 Jensen 的 CPN 定义,这主要是因为 DesignCPN 工具能够全面支持和接受它,以方便编辑、仿真和校验 CPN。

2　方法介绍

在这一节中,我们详细说明怎样使用 UML 和 CPN 从概要导出 UI 的原型系统。图1给出了活动的内容和顺序。

在概要获取活动中,分析员精确给出每一个用例的用例图和相应的序列图。规格说明建立活动是从所获得的用例图和序列图导出系统的 CPN。概要集成活动是将相同用例相对应的 CPNs 交互式地合并成集成 UI。集成 UI 作为 CPN 验证和原型生成活动的输入。在原型的评估活动中,所得到原型应由最终用户来执行、来评价。在 CPN 验证活动中,用现有的算法来检测其行为特性。

在下面几小节中,我们将详细讨论 UI 原型开发过程中的三个活动:概要的获取、规格说明的建立、概

魏定国　副教授,博士生,主要从事网络协议与分布式软件、数据库的研究,**吴时霖**　教授,博士生导师,主要从事计算机网络协议与分布式软件、Petri 网理论及应用的研究。

要的集成和 UI 原型的生成。而 CPN 验证限于篇幅就不在此讨论。

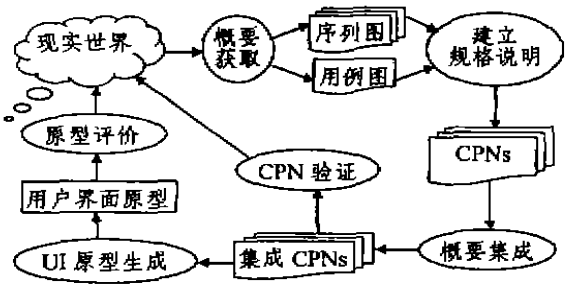

图1 原型设计过程

2.1 概要的获取

在这个活动中，分析员详细说明用例图以捕捉系统的功能特性，对每一个用例分析员应当给出相应的以序例图形式表示的概要。

一个给定用例的概要是可以按类型分类，并能按应用的频率排序。我们来考虑两类概要：正常概要，它是正常情况下执行；另一类异常概要，它是在错误和异常的情况下执行。概要使用频度（或执行频率）由分析员定义为10个等级，用来说明给定概要的发生频度。例如用例标识有一个正常的概要（概要 regularIdentify 频度为10）和异常的概要（概要 errorIdentify 频度为5）。

2.2 规格说明的建立

这项活动就是要从所获得的用例图和序列图导出 CPNs 的这个导出过程，下面我们用用例规格说明和概要规格说明来解释。

2.2.1 *用例规格说明* 将用例映射到位置就可导出与用例图相对应的 CPN。这个转换导致由一个位置（Enter）与用例的初始动作相对应。开始位置（Begin）总是加到系统的初始状态的模型上，在一个用例执行之后系统将会返回，通过一个退出转移（Exit），返回到它的初始状态以便其它用例的执行。开始位置可以包含几个标志来完成多个并发执行的建模。

在一个用例图中，用例能够通过关系应用调用其它用例的服务。考虑用例 Uc_1 使用用例 Uc_2，图2(a)中给出了关系的一般形式。用例 Uc_1 可以分解为三个子用例：Uc_{11}，表示在调用 Uc_2 之前的执行部分；Uc_{12}，表示与 Uc_2 并发执行的部分，而 Uc_{13}，表示在 Uc_2 执行完终止以后的执行部分。应该注意到这三个部分中可能有一两个为空。图2的(a)～(g)详细说明了八种可能的映射。在 Uc_1 和 Uc_2 之间的类型(g)关系意味着 Uc_2 先于 Uc_1。这也就表明了不能直接从开始位置接近 Uc_1。因此开始位置到表示 Uc_1 的位置转移必须用 Enter 进入 Uc_2 的 Enter 转移和从 Uc_1 进入 Uc_2 的 Enter 转移来代替。

在 CPN 设计工具 DesignTool 中，允许转移逐步求精，但不支持位置求精。因此为了代替这些表示位置的用例，对描述集成概要的 CPN，在处理应用关系之后需要作一些适当调整：对 CPN 的每一个子集 Enter →place$_i$→Exit 用一个简单的表示用例的转移来代替。

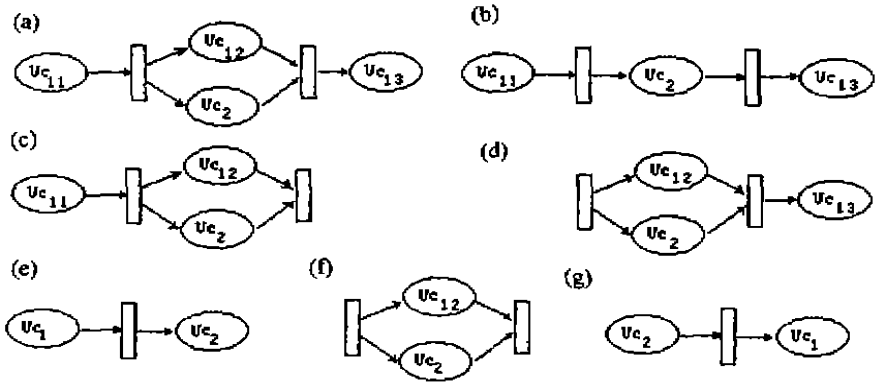

图2 应用关系的各种可能映射

2.2.2 *概要说明* 对每一个给定的用例，分析员建立一个对象的关联表，这个表直接从概要的序列图获得，从顶到底交换消息，并标识由这些消息引起的对象状态的变化。在这些表中，概要状态是由参入对象的状态矢量来表示。

从每一个对象状态表，通过概要状态转换成位置、消息转换成转移就可以产生 CPN。每个概要安排一个不同的颜色。所有相同用例的 CPNs（概要）将有一个初始位置（状态）。这个位置将通过系统的用例图建模所得的 CPN 链接到集成的 CPN 上。

可以看到，在概要说明中，只有对象状态表是手工建立的，其余部分都是完全自动的，这种方法的下一步

是概要的集成，需要输入串行化的CPN。因为CPN设计工具是用SGML作为交换格式，在SGML、XML和Java之间的转换工具是很好用的，所以可以用XML来表达CPN。使用designCPN设计工具，分析员可以方便地编辑、保存用例CPN，所有的概要CPNs可以方便地转移成SGML文本格式。然后使用SX工具将SGML转移成粗略的XML(RXML)，再用Xjparse Java程序将RXML转移成XML。

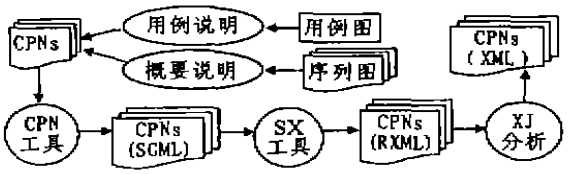

图3 从用例图和序列图生成XML格式的着色Petri网

2.3 概要集成

在这项工作中，为了产生一个集成的CPN给用例的行为建模，必须合并所有的与用例Uc_i的概要相关的CPNs。我们提供一个算法，采用递增的方法进行集成。对给定的两个与CPNs相关的两个概要CPN_1和CPN_2，该算法能够合并在CPN_1和CPN_2中具有相同名称的所有位置。两个概要的颜色的并作为合并位置的颜色。算法再在两个概要中寻找具有相同输入和输出位置的转移，并在它们的条件之间用一个"或"(OR)来合并它们。下面我们用伪代码来描述这个算法：

```
Uc1. IntergrateScenarios()
    scList = Uc1. getListofScenarios();
    //返回用例Uc1的概要的一个列表
    uc_cpn=getXML(scList[0]);
    //返回与概要scList[0]相对应的XML文件
    i=1;
    while (i<scList. size())
        sc_cpn=getXML(scList[i]);
        sc_cpn=makeUniqueID(uc_cpn,sc_cpn);
        uc_cpn=merge(uc_cpn,sc_cpn);
        i=i+1;
    end
end Uc1. IntergrateScenarios
```

在CPN(位置、转移、边等)中，XML对每一元素都用不同的标识符(ID)来标识。在合并两个概要之前，方法makeUniqueID检查两个输入文件uc_cpn和sc_cpn，看是否有不同的IDs。在它们共享共同的IDs情况下，makeUniqueID修改sc_cpn的IDs，将uc_cpn的最大ID加到sc_cpn的IDs中。

合并(或集成)两个概要，其CPNs分别具有颜色$[sc_1]$和$[sc_2]$，将产生以表$[sc_1, sc_2]$作为颜色的CPN，这个合并操作用下面的算法来描述。

```
merge(uc_cpn,sc_cpn)
    uc_cpn. addPlaces(sc_cpn)
    //将那些uc_cpn不存在的sc_cpn的位置加入到uc_cpn中
    for each t in sc_cpn getListOfTransition()
        t'=uc_cpn. LookForTrans(t)
        //t'是一个满足't='t'和t'=t''的一个条件转移
        if (t' does not exist)
            uc_cpn. addtrans(t)
        endif
    end
    uc_cpn. addEges(sc_cpn)
    //将那些在uc_cpn中不存在的sc_cpn的边加到uc_cpn中
    uc_cpn. mergeColors(sc_cpn)
    //给集成的CPN(uc_cpn)计算新的颜色
    uc_cpn. putColorOnPlaces(sc_cpn)
    //CPN网的所有位置着上合并的颜色
    uc_cpn. putGardOnTransition(sc_cpn)
    //合并的颜色将守卫着公共转移，其它将由它们原来的色守卫着
    uc_cpn. putVariablesOnEdges(sc_cpn)//在边上安置变量或标记表达式
end merge
```

2.4 用户界面原型的生成

在该活动中，我们可以从CPN规格说明中导出系统的UI原型。生成的原型是独立的，组成一个菜单用于不同用例之间的切换。原型的各种画面表达了UI的静态特征。在CPN规格说明中所捕获的UI动态方面，将映射成原型的对话控制。原型是Java应用程序，由几帧画面组成，具有导航的功能特性。

原型生成的活动包括：生成转移图GT；屏敝所有非交互转移；标识UI块；构成UI块；从构成了的UI块生成界面画面帧。这些操作基本上可以与基于状态图方法相对应，只有生成转移图除外。这些操作是由从给定用例的CPN导出的有向转移图(GT)所组成的，CPN的转移将代表GT的结点，边指明转移执行优次序：如果两T_1和T_2连续执行，那么在表示T_1和T_2的结点之间就有一条边。

GT有向图有一个结点表nodeList、边表edgeList和初始结点表initialNodeList(进入图的结点)，GT的结点表nodeList很容易获得，因为相应的CPN的转移表是现成的，GT有向图的边表edgeList可以在CPN中对每个位置p从联接'p与p'的转移获得。

3 方法的讨论与实现

该UI生成的方法能够充分发挥基于概要技术的优点。与很多面向数据的方法相比，UI是从描述系统的动态行为的规格说明中产生，而规格说明是从任务分析中导出的。一旦形成规格说明，就可以将其用作进一步精化的基础，而且新概要可以从已有的概要中产生。我们认为，应该注重的是动态模型方面和生成UI动态的核心部分，而不是屏幕设计和用户系统的对话。

众所周知，当处理一个大型的应用时，可扩展性是

(下转第77页)

3 Flanagan T. Fiber Network Survivability. IEEE Communications Magazine, June, 1990 46～53
4 OPTICAL RESTORATION: 1, OFC'98. VOL 2, PP. 269～301
5 Demeester P, Gryseels M, et al. Resilience in Multilayer Networks IEEE Communications Magazine, August, 1990. 70～76
6 Grover W D. The Selfhealing Network-A Fast Distributed Restoration Technique For Networks Using Digital Cross-connect Machines, Proc., IEEE Globecom'87, 1987, pp. 28. 2. 1～28 2 6
7 Yang C H, Hasegawa S. Fitness Failure Immunization Technology For Network Service Survivability, Proc., IEEE Globecom'88, pp. 1549～1554
8 Kome H, et al. A distributed restoration algorithm for multiple-link and node failures of transport networks, proc., IEEE Globecom'90, San Diego, CA, December 1990. 459～463
9 Sakauchi H, Nishimura Y, Hasegawa S. A Self-Healing Network With An Economical Spare-Channel Assignment, Proc., IEEE Globecom'90, San Diego, Ca, Dec. 1990. 438～443
10 Coan B A, et al. Using Distributed Topology Update And Preplanned Configurations To Achieve Trunk Network Survivability. IEEE Transactions On Reliability, 1991, 40 (4). 404～416
11 Grover W D, Billodeau T D, Venables B D. Near Optimal Spare Capacity Planning In A Mesh Restorable Network, proc IEEE Globecom'91, 1991. 2007～2012
12 Anderson J, et al. Fast Restoration of ATM Networks. IEEE JASC, 1994, 12(1), 128～137
13 Doshi B T, Dravida S, Harshavardhana P, et al. Optical Network Design and Restoration. Bell Labs Technical Journal, January-March, 1999 59～84
14 Ye Yinghua, Dixit S, Ali M. On joint Protecting/Restoration in IP-centric DWDM-based Optial Transport Networks. IEEE Communications Magazine, Jun. 2000. 174～183
15 Zolfaghart A, Kaudel F J. Framework for Network Survivability Performance. IEEE JSAC, 1994. 12(1): 46～51
16 Sosnosky J. Service Applications For SONET DCS Distributed Restoration. IEEE Journal on Selected Areas in Communications, 1994, 12(1). 59～68
17 Gerstel O, et al. Combined WDM and SONET Network Design. Proc. INFOCOM'99, 1999, 2: 734～743
18 Herzberg M, Bye S J, Utano A. The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks. IEEE/ACM Transactions on Networking, 1995, 3(6). 775～784
19 Falconer W E. Service Assurance in Modern Telecommunications Networks. IEEE Comm. Magazine, June, 1990
20 Wu T H, et al Survivable Network Architectures for Broad-Band Fiber Optic Netwroks: Model and Performance Comparison Journal of Lightwave Technology, 1988, 6(11). 1698～1708
21 Bellary A, Mizushima K. Intelligent Transprot Network Survivability: Study OF Distributed And Centralized Control Techniques Using DCS & ADMS. In, Proc. IEEE GLOBECOM'90, 1990. 1264～1268
22 Ramamurthy S, Mukherjee B. Survivable WDM Mesh Networks, Part II-Restoration. Proc. ICC'99, 1999, 3: 2023～2030
23 Kawamura R, Sato K, Tokizawa I. Self-Healing ATM Networks Based on Virtual Path Concept. IEEE JSAC Special Issue, Integrity of public Commnu. Networks, 1994, 12 (1). 120～127
24 Venables B D, et al. Two Strategies for Spare Capacity Placement in Mesh Restorable Networks. In: proc. ICC' 93, 1993. 267～271
25 Nederlof L, Struyve K, et al. End-to-End Survivable Broadband Networks. IEEE Comm. Mag., 1995(Sep): 63～70

(上接第109页)

一个很关键的问题，此方法能很好地解决这个问题，因为它是以每个用例为基础来集成概要，而不是将它们直接当作一群无结构化的概要。

概要集成和原型的生成活动都可以用 Java 来实施，大约需要4,000行代码。UI 原型的 Java 代码与界面开发环境 Visual Cafe 完全兼容，可以直接嵌入到可视化环境中。

结论 本文提供了基于概要的用户界原型开发的一种新的方法，只要序例图附上丰富的 UI 信息就可以获得系统的概要。将序例图转换成 CPN 说明，不论 UI 是静态的还是动态的行为特性都可以从 CPN 规格说明中得到。从此说明就可以生成系统的 UI 原型。所获得的原型提供给最终用户进行有效性验证，逐步接近其目标。

参 考 文 献

1 DesignCPN: Version3 Meta Software Corp. http://www.daimi.aau.dk/~designcpn
2 Booch G, Rumbaugh J, Jacobson I The unified modeling language user guide: The ultimate tutorial to the UML from the original designers. Addison-Wesley object technolgy series. 1999
3 Kristensen L M, Cristensen S, Jensen K. The Practitioner's Guide to Coloured Petri Nets. Softeware Tools for Technology Transfer, 1999. 2(2): 98～132
4 IBM. XML Parser for Java in Java Report's Feb. 1999. http://www.alphaworks.ibm.com/formula/xml
5 Elkouti M, Khriss I, Keller R K. Generating User Interface Prototypes from Scenarios. In: Proc. of the Fourth IEEE Intl. Symposium on Requirements Engineering(RE' 99). Limetick, Ireland, June 1999. 150～158
6 Smantec, Inc. Visual Cafe for Java, User Guide, Symantec, Inc. 1997