面向可用性的软件认知过程建模与模式检查方法*)

赖祥伟　邱玉辉　张为群

(西南大学计算机与信息科学学院　重庆 400715)　(重庆市智能软件与软件工程重点实验室　重庆 400715)

摘　要　软件可用性检查是满足用户体验,提高软件质量的重要方法之一。本文提出了使用 Markov 随机过程模型建立了用户交互过程认知模型的基本方法,并在此基础上给出了一个严格的可用性模式检查过程和方法,帮助软件开发者重构用户交互过程,为改善软件系统的可用性特征提供了一种有效的手段。实践证明,该模型对不同类型用户具有很强的区分能力,能够有效地体现用户操作的概率特征,改善了传统可用性方法过分依赖可用性专家个人能力的缺陷,对于各种不同类型的软件系统具有较好的通用性特征。

关键词　软件可用性,Markov 随机过程模型,模式检查,STNs

Usability-oriented Interaction Process Cognitive Process Modeling and Model Check Method

LAI Xiang-Wei　QIU Yu-Hui　ZHANG Wei-Qun

(Faculty of Computer and Information Science, Southwest University, Chongqing 400715)

Abstract　Usability evaluation is increasingly being used in the development of software. Users always get intuitionist feeling to a product by the human-computer interaction process. This paper we present a usability-oriented interaction process rebuild method. A Markov model is used to modeling the human-computer interaction process. The Markov model can be transformed to an operation process diagram by a serial of strict steps. Some model checking rules were suggested to monitor and improve the interaction process. And we also get an effective reflection either.

Keywords　Usability, Markov random process, Model check, STNs

1　前言

一般认为,用户的软件使用过程是一种典型的自我满足过程。在这一过程中,用户总是不断尝试各种不同的技术路线和方法,以期达到预期的结果。普通用户评价软件的唯一标准是该软件是否能够提供给他们达到预期结果的稳定方法。然而,随着软件开发技术的日益发展,软件用户开始对软件完成制定任务的时间、操作过程乃至使用感受提出了更多的期望和要求。对于某些"特殊"客户(如企业信息系统管理者、软件设计师等)而言,系统的能力和效率已经不能再满足他们对于系统可用性特征的要求。他们的更多注意力开始集中于软件用户是否能够尽快无障碍地学会软件的使用方法,甚至于开始关注用户在软件使用过程中是否一直处于愉悦的状态中等进一步的软件可用性特征。这些都使得软件可用性工程日益为开发者所接受,并被广泛应用于软件开发过程当中。

研究表明,除了系统功能度和可靠性以外,软件的可用性特征是软件成功最为重要的因素[1]。成功的软件可靠性设计需要考虑人机交互、GUI、工程心理学、人类功效学等多方面的因素[2,3]。广义地讲,软件可用性取决于用户在特定使用环境下的用户感受。用户的个体差异、任务的具体特征、所处的社会环境和物理环境都将直接影响用户的可用性感受。在ISO9126[4~6]中,软件可用性定义为软件在通常情况下易学习、易使用和被用户喜爱的特征。由此可见,软件可用性是软件系统的能力、效率以及用户满意程度的综合体现。

认知模型[7]是一种对用户行为的计算机模拟。通常意义下泛指使用类似于人类心算的方法对于各种复杂实时系统进行模拟控制[8~0]。要实现自然的人机交互离不开认知心理学的支持,要充分利用人类的认知能力,建立一个人机能够相互理解和获取信息的系统,则需要人机交互中的有关理论、模型和算法都必须符合人类认知特点。Ritter and Major(1995)认为认知模型完成用户任务的方法包括如下 3 种:

• 使用认知建模语言的交互任务模拟建模方法研究。该方法适用于描述简单的认知过程内在关系。当前已有众多的试验模型(如 Beaman and Morton[1998], Newell and Simon [1972], Peck and John[1992], and Ritter et al. [1998])。

• 使用模拟语言的用户任务描述工具研究。早期的版本如 EPIC[Kieras and Meyer 1997], Ritter et al. [1998] and Taatgen[1998]等。

• 交互任务模型与用户模型结合的任务分析方法(如 Gray[2000])。该方法适用于已有交互模型的进一步分析,特别是针对某些复杂任务的交互过程分析具有很好的效果。

我们一直在尝试用户交互过程的建模方法研究。本文给提出了一种面向软件可用性的系统模式检查,重构软件交互过程的系统方法。该方法的主要目的在于为软件可用性改进建立理论基础。当前的主要工作是交互过程的模型建立方法和基于模型的可用性验证技术。

2　软件交互过程模式检查框架

如图 1 所示,我们给出一个用于改善软件可用性特征的用户交互过程模式检查框架。

*)本文受到重庆市自然科学基金重点项目"软件测试技术与方法研究"支持;重庆市西南师范大学青年基金项目(SWNUQ2005011)(SWNUQ2005020)。**赖祥伟**　博士生,讲师,CCF 会员,主要研究领域为软件测试、智能计算技术;**邱玉辉**　教授,CCF 会员,主要研究领域为人工智能;**张为群**　教授,CCF 会员,主要研究领域为软件工程。

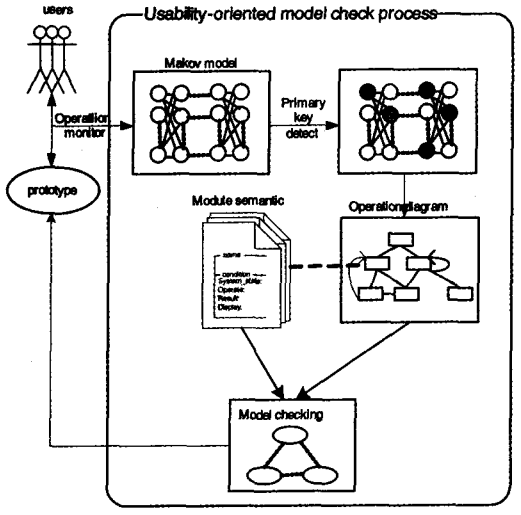


图 1　交互过程模式检查框架

该框架主要实现了如下功能：

· 使用 Markov 随机过程模型对用户交互行为进行建模；

· 使用 Markov 转移概率矩阵提取软件关键操作和关键路径；

· 根据关键操作和关键路径构造系统交互状态转移网络图(STNs)；

· 对状态转移网络图的关键节点进行形式化描述；

· 根据系统可用性模式检查规则对系统进行模式检查，发现潜在的可用性问题；

· 重构交互过程并进行进一步的循环检查。

本文将具体讨论该框架的实施方法和细节。

3　用户操作认知模型

用户软件体验过程实际上是一个典型的随机过程。由于环境和认知能力的影响，用户总是在各种可能的系统功能间游走，并根据主观判断与系统进行各种成功和不成功的交互活动。在对于这一过程的研究过程中，我们发现一个随机 Markov 过程模型对该问题有很好的表现作用。

3.1　用户操作的 Markov 过程模型

使用者的操作过程被看作是一个典型的非连续变化的随机过程[7]。每个操作的结构可以近似地认为只与其上一个操作相关(在实际问题中可能存在例外的情况，但为了处理方便，这里做了这样的假设)。这个随机过程可以定义为

$$P(\xi_{n+1}=j \mid \xi_n=i_n, \xi_{n-1}=i_{n-1}, \xi_{n-2}=i_{n-2}, \cdots, \xi_0=i_0)$$

其中 ξ_n 指用户在时间 n 时的操作行为。

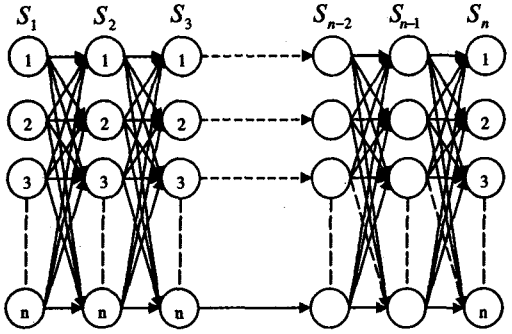


图 2　用户操作过程 Markov 过程模型

根据软件使用的特性可以认为所得到的 Markov 模型具有如下的性质。

[性质 1]：主界面对应的状态是 Markov 链中的一个常返态。

[性质 2]：除主界面操作外的所有状态可以被认为是一个暂态。

[性质 3]：每个使用者的操作序列是一个时齐的 Markov 链。

[性质 4]：所有对于主界面的操作可以分解成为两个相连的操作：①返回主界面；②选择执行主界面上相应的功能。

该 Markov 随机过程模型的转移概率可以由多个用户操作的监控和简单统计方法获得。常用的统计公式如下：

$$t_{ij}=\begin{cases} num_{ij}/num_{all} & ((1\,000\times n\times num_{ij})>num_{all}) \\ 0 & ((1\,000\times n\times num_{ij})\leqslant num_{all}) \end{cases}$$

其中 num_{ij} 表示从状态 i 到状态 j 的转移次数。num_{all} 表示用户所有的操作次数，如果 $(1\,000\times n\times num_{ij})\leqslant num_{all}$，则可以认为状态 i 和状态 j 之间没有可能的转移可能，而个别的 i,j 转移则被认为是用户的偶然操作或错误操作而加以忽略。由此得到的系统转移概率矩阵可以定义为

$$M_T=\begin{bmatrix} t_{11} & t_{21} & t_{31} & \cdots & t_{n1} \\ t_{12} & \cdot & \cdot & \cdot & t_{n2} \\ t_{13} & \cdot & \cdot & \cdot & t_{n3} \\ \vdots & \vdots & \vdots & t_{ij} & \vdots \\ t_{1n} & \cdot & \cdot & \cdot & t_{mn} \end{bmatrix}$$

通过对特定模型转移矩阵的状态转移概率的判定可以得到当前模型的特征转移路径。核心路径的判断方法为

$$MP_{ij}(n,m)=\bigcup_k P_{x(x+1)}(s,e) \mid ((k=m-n)\wedge(x\in[n,m-1])\wedge(\forall x\cdot E_{x,(x+1)}\geqslant\max(E_{x,j})))$$

由核心路径的计算方法可知，由此得到的核心路径是使用者在使用软件完成某指定任务时最为常见的操作过程。该操作过程能够有效地反映该类使用者的操作特征，这种特征的确定对于分析软件针对特定使用者的可用性特征具有重要的作用。

所有计算所得的关键节点和关键过程都将在 Markov 模型中标记出来(如图 3)。

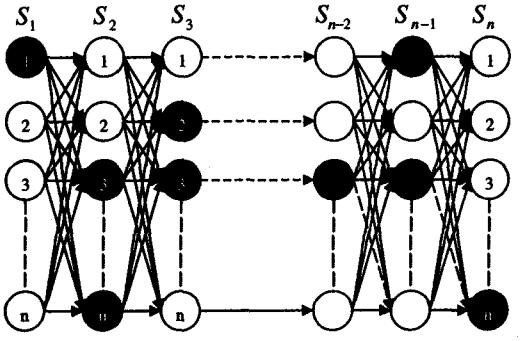


图 3　标记核心路径的 Markov 模型

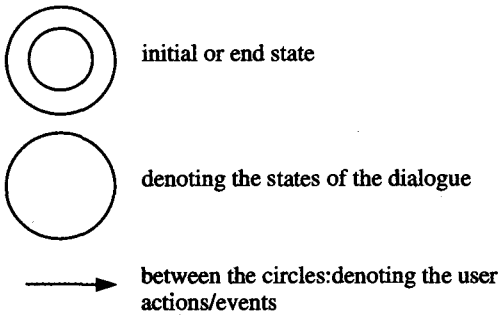
我们采用传统的 Markov 解码算法以预测用户的可能路径。可能路径能够很好地描述可能的用户操作过程。软件开发者可以通过对于关键路径的分析了解用户的操作习惯和软件的用户操作特征。常用的关键路径分析算法如下：

```
Procedure primary_road(a,b)
//Where a_j,b_j are the transform probability on the time of j
begin initialize Path←{},t←0
    for t←t+1
        j←1
        for j←j+1
            a_j(t)←b_jk v(t) Σ_{i=1} a_i(t−1)a_ij
        Until j=c
        j' ← arg max a_j(t)
                 j
```

```
        Append w_j to Path
    Until t=T
    Return Path
end
```

3.2 Markov 模型的 STNs 转换

关键路径有效地体现了用户的主要操作过程和软件使用特征，这些特征是改进软件可用性的重要依据。对于关键路径的关注是开发者应当逐步重视的问题。由于使用 Markov 模型标记的关键路径过于抽象，不利于普通开发者分析和使用，我们采用系统状态转换网络(STNs)来对系统关键路径做进一步的表示和分析。系统状态转换网络(STNs)的主要符号集合如下：



标记关键路径的 Markov 标记模型，其 STNs 转换算法如下：

```
Procedure Markov2STNs
Begin
// confirm "initial states" and "final states"
For (i=1 to n)
    {initial=true; final=true
      For (j=1 to n)
          {
        If t_{j,i}=0 then initial=false;
        If t_{i,j}=0 then final=false;}
      If initial=true then signed S_i as initial state
      If final=true then signed S_i as final state}
//construct the operation tree
Traversal all the states, to each states
    {if (S_i∈MP) ∧ (S_j∈MP) ∧ (t_ij≠0) then add_arrow(i,j);}
End
```

3.3 STNs 的形式化描述方法

STNs 是一种有效的人机交互过程描述方法。但是，由于其图形化特征导致它对状态的操作语义的规约能力较差，为了弥补其缺点，我们给每个关键状态定义了一种类似于 Z 语言的操作语义定义规范(图 4)。这些操作语义描述主要聚焦与该状态的潜在操作语义，对于状态约束和彼此相关约束具有很好的描述能力。

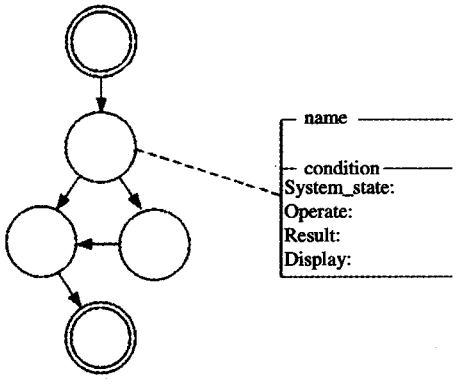


图 4　STNs 的形式化描述方法

STNs 的符号集合需要给出相应的定义[13]。用户的所有输入被视为一个操作。所谓操作可以描述系统不同粒度的用户行为，既可以指用户的某个单独击键操作，也可以定义用户对于某个电子表格的一系列动作。对于用户系列行为的操作可以定义为

$C=\{move\_right, \cdots, insert\_col, set\_formula(\text{“}A1+B1\text{”}), set\_formula(\text{“}23\text{”}), \cdots\}$

操作集合“”中的部分同样可以分解成为几个粒度更低的操作：

$set\_formula(\text{“}AB\text{”})=\{set\_formula(\text{“}A\text{”}), set\_formula(\text{“}B\text{”})\}$

$where\ set\_formula(\text{“}B\text{”})\ atnext\ set\_formula(\text{“}A\text{”})=t$

我们给出用户操作的归纳定义：

- 所有原子操作可以被定义为一个操作；
- 如果 A 是一个操作，那么 $\neg A$，$\circ A$ 和 $\Box A$ 也是一个操作；
- 如果 A,B 是一个操作，那么 $(A\rightarrow B)$ 以及 $A\ atnext\ B$ 可以被认为是一个操作。

STNs 模型将系统显示与交互结果区别对待，虽然它们都是系统的输出。系统环境与输出的数学描述为

$display: E\rightarrow D$

$result: E\rightarrow R$

其中 R 指系统的潜在运行结果，D 指系统的显示输出，E 是系统运行环境和条件。

当前状态 E 是指系统所有历史操作的集合，系统的历史行为可以定义为

$P=seq\ C$

$I: P\rightarrow E$

4　交互过程的模式检查规则

对不同状态的 Markov 模型的定义是后续工作的重要基础。在利用已有的试验数据初步确定 Markov 转移矩阵的参数后，这些模型对于使用者操作情况有很强的描述能力。在此基础上的应用包括如下几点：

- 首达率指标
- 常返态指标
- 状态回访率指标
- 操作条件语义

4.1　状态首达率指标

对于任何当前使用者，可以利用状态空间某指定状态的首达率判断使用者的状态和情况。在 Markov 模型中，指定状态的首达率定义为

$$f_{ij}^{(n)}=P(\xi_n=j,\xi\neq j,k=1,2,\cdots,n-1|\xi_0=i)\quad (n\geqslant 1)$$

若使用者从状态 i 出发到底状态 j 的实际操作步骤与两状态间的首达率存在较大的差异，可以认为使用者在完成该功能过程中遇到了一定的困难。这时，相应提示能够有效地帮助使用者完成相应的工作，避免过多的时间浪费和对使用者的使用情绪产生影响。

4.2　常返态访问概率指标

根据 Markov 理论，设常返态 i 的首达时间为 T_0，以后的各次返回时间间隔为 $T_1,T_2,\cdots,T_n$，可证$\{T_n: n\geqslant 1\}$是独立分布的，而且与 T_0 独立，由此生成一个延迟更新过程 $N_i^{(i)}$。T_i 的分布为 d-格点分布，其中 $d\geqslant 1$。显见：在 $d\geqslant 2$ 时，d 就是状态 i 的周期。

根据常返态极限定律，可以求出该常返态的平均回访概率 S_a。如果$|S_i-S_a|\leqslant 2e$(S_i 为在最近一次常返态访问后的操作步数)，则可认为使用者处于正常操作过程，反之则认为使用者在当前操作中遇到困难，应根据帮助规则模型为用户提供下一状态的指导。

4.3 状态回访率指标

一些状态的平均回访时间可用于判断使用者的无助和困境。

对于某常返态的较少回访时间往往是由以下两种可能引起的：

(1) 使用者操作顺利，操作效率较高(软件设计合理，可用性强)。

(2) 使用者遇到较大的困难，不断尝试各种可能的途径，但多次碰壁，被迫返回。可以使用下列公式来判断实际情况到底属于上述哪种情况：

$$f(n)=1\Leftrightarrow\exists\, l\cdot p_{ij}(n,n)=\sum_k p_{ik}(n,l)\,p_{kj}(l,n)$$

该公式表明，如果使用者在回访某状态前曾经访问过利用此式提取的 primary stations。则可以认为该状态的回访是一个正常的回访过程。否则，该转态的回访应被认为是使用者操作可能遭遇困境。该方法虽然在理论上缺乏完整性(在正常操作过程中确实存在不经过 primary stations 而直接回访某状态的可能性)，但试验表明在实际使用过程中，该方法存在极强的描述能力。

4.4 状态操作语义

良好的操作语义对开发和测试都起着至关重要的作用[14]。从测试的角度来看，与测试用例生成相关的类定义操作语义主要包括 3 个方面。

前置条件：前置条件是当操作被执行之前应该满足的条件。它通常是根据以下两种情形来描述的：一种是包含操作的对象的属性，另一种是请求执行这一操作的消息中包含的实际参数的属性。

后置条件：后置条件是当操作被执行结束后所满足的条件。它通常根据以下几点来描述：

- 包含操作的对象的属性变化情况；
- 请求执行操作的消息中所包含实际参数的属性的变化情况；
- 根据任意返回值来决定；
- 根据可能出现的异常来决定。

在类中，所有操作说明的集合对类实例的行为做了部分的描述。行为很难单独从操作中推断出来，所以行为通常是经过状态转换的更抽象的表现形式。行为被形象地定义为实例的一系列状态，而且还描述了各种操作怎样影响状态之间的转换。

相关性条件：相关性条件是后置条件的一种特例，主要指操作之间或操作与属性之间的相互关联条件，其基本规则是一个操作的结果不能够对其它操作产生不良影响。因为它在测试过程中有重要作用而被单独定义。

操作语义对于操作行为和结果的影响是可以预测的，这正是基于操作语义的测试用例构造的基础。使用前置条件和后置条件结合生成测试用例的算法如下：

```
procedure condi_combin
    begin
    set pre={};
    set post={};
    pre = combin (conditions); //add the possible pre-condition
                               combination
                               //into the pre by the table 1;
    post = combin (conditions); //add the possible post-condition
                               combination
                               //into the post by the table 2;
    check-model= pre × post //make the set check-model as the
                            Descartes Set
                            //of pre and post. ;
    if (the condition combination is nonsensical)
        then delete the combination from the check-model set.
end
```

5 试验和结果分析

我们采用上述方法对一个非常通用的在线邮件系统进行用户交互过程建模和交互过程改进，以验证本方法的有效性和实际效果。正如多数人看到的那样，当前的在线邮件系统已经是一种非常成熟而通用的软件系统了，用户在使用类似系统过程中已经养成了许多固有的认知习惯。但是令人惊讶的是，对于这样的系统，验证结果依然存在许多系统潜在的可用性问题。

共有 30 名用户参加了指定系统的使用体验活动。其中三分之一的用户从未使用过类似的系统，三分之一的用户有过类似系统的使用经验，而最后三分之一用户是经验丰富的系统使用者。为了帮助有需要的用户顺利完成体验活动，组织者在体验活动过程中可以对用户的疑问进行一定的解释和指导，但这些意见被要求尽可能简单并不带有命令性色彩。

所有的用户交互活动被一个屏幕录制软件所记录，以提供后期分析所需的基本数据。经过统计发现，合计有 3274 次用户交互行为被承认为合法的记录。其中 14 种具有最高操作概率的行为被最终选择用户构建用户的系统交互 Markov 模型，它们覆盖了所有用户操作的 74.6%。由此得到的用户交互行为转移矩阵如图 5 所示。

$$\begin{bmatrix}
0 & 0.0302 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.0437 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.0339 & 0 & 0 & 0.0125 & 0 & 0.0282 & 0 & 0 & 0 & 0.0358 & 0 & 0 & 0 & 0.0292 \\
0 & 0 & 0 & 0.0649 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0416 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0473 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0381 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0428 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0252 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0710 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0152 & 0 & 0
\end{bmatrix}$$

图 5 14 种主要用户交互行为的状态转移矩阵

从图 5 中我们可以发现，状态 3,7,8,13 与其它状态之间的转移概率较低(根据公式被忽略记为 0)。使用前文中的 Makov2opraform 算法可以得到系统的 STNs 图(图 6)。由于篇幅限制，我们略去相关状态之间的转移条件和每个状态的形式化描述。

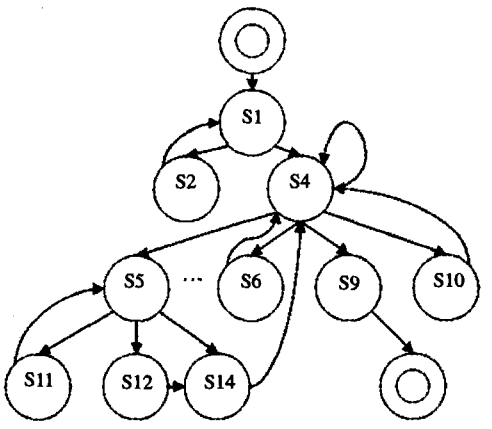


图 6 系统 STNs 图

采用前文中的模式检查方法，我们发现了一系列的系统可用性潜在问题。例如：

(1) 通过对于状态转移模型的分析，我们发现从 state(new_message) 到 state(drafts)之间有很高的状态转移概率。通过屏幕录像软件录像分析发现，对应用户行为是：

Click_from("To:bai"), click_menu(back)

访谈参加体验的用户，发现导致该回退行为的主要原因是用户希望能够给用户某邮件的发送者"bai"回复邮件。当用户发现界面中有"to bai"显示时，用户自然认为该按钮的主要功能是给用户"bai"发送邮件。但是，当用户点击该链接并进入下一界面时，发现所产生的邮件的收件人却是自己(如图7所示)。当用户发现操作结果和预期结果不符合时，自然选择了回退操作。

导致该问题的主要原因是在图7上方邮件信息中显示的"to:bai"链接实际上表示的是邮件的来源(所以放在"from"标题下)。而错误的软件设计者将"from:bai"错误地表示为"to:bai"从而导致用户的理解错误。

(2)使用前文中描述的条件组合模式检查方法，我们还发现了一个较有代表性的问题。在通常设计理论下，用户可以通过两条转移路径到达状态 drafts:其一是在 state(folder_tree)时执行操作 click(draft)，另一种是使用下拉菜单中的"open_folder"操作(如图8所示)。

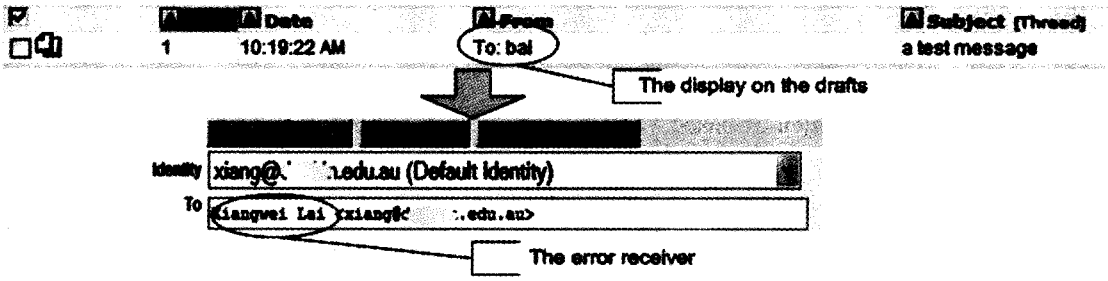


图7 问题1示意图

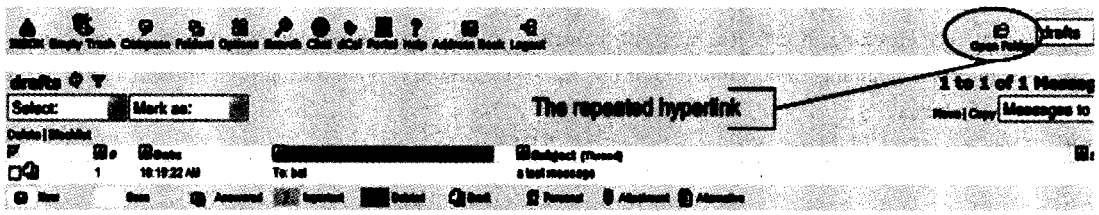


图8 问题2示意图

以下为图8中状态 drafts_list 的形式化描述：

name

Drafts_list

condition

System_state:
username=***
Operate:
(state("folder_tree")∨click(folder_tree.drafts))∧
(drop_ment("drafts")∧
((drop_ment("drafts")∨(click(hy_"open_folder")atnext
(drop_ment("drafts")))
Result:
(state_tras(*,drafts_list))
Display:
(show(drafts_list))

对于该状态的条件组合包括如下两个：

-(username= * * *)∧(drop_menu("drafts")

-(username = * * *)∧((drop_menu("drafts")∨(click(hy_"open_folder") atnext (drop_menu("drafts")))

⇒(username= * * *)∧(drop_menu("drafts")

通过对于两个条件的分析，我们发现操作：(click(hy_"open_folder") atnext (drop_menu("drafts")是一个冗余的条件，这就意味着链接"open_folder"是一个无效的链接。

(3) 另一个典型的问题出现在状态 trash 的使用过程中。当用户创建了一个叫做"trash"的邮件夹后，在系统的邮件夹列表中并没有出现相应的邮件夹名称和图标，许多用户因此认为系统出现问题。实际的情况是，相应的邮件夹名次和图标必须在手工刷新页面后才能出现(如图9所示)。

上述问题可以认为是一个系统设计者的缺失，因为他们忘记了在用户新建邮件夹工作完成后让系统自动执行刷新操作。这样的问题通过对于操作后置条件的检查可以很好地发现。设计者原有的操作条件组合为

Oeration(new_folder)→new_folder(foldername)

正确的条件组合应该是

Oeration(new_folder)→new_folder(foldername)∨operation(refresh)

结论和未来工作 必须指出，本文研究的主要目的在于从可用性的角度提高软件系统的质量。试验的参与者均依赖个人对软件系统的认知，并未受过专门的训练和培训[15]。因此，他们的使用过程可能并未覆盖软件系统的所有功能。同时，本文提出的方法也需要在其它的软件系统平台上进行试验。我们认为，该方法的相关经验能够在各种类型的系统开发组织中得到有益的尝试和使用，并帮助系统开发者在改进软件可用性改进方面作出一定的贡献。

我们认为在后续工作中还有许多方面的工作需要努力，它们包括：

- 进一步完善用户操作的形式化描述符号体系。
- 开发基于该模型的系统自动模型检查工具集。
- 在人机交互过程建模中引入多通道技术，增强用户模型的刻画能力。

我们同时希望为其它软计算技术在软件过程改进中的应用做出其它有益的尝试和努力。

(下转第279页)

预测值与观测值的相对误差的方差为 18.67，相对误差的方差小说明预测较稳定！

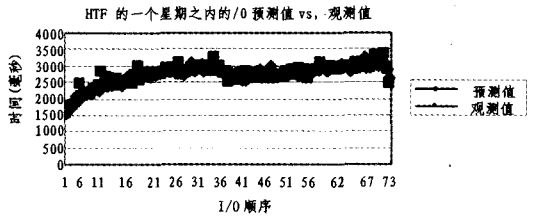


图 2　HTF 的一个周期之内的 I/O 序列的观测值与预测值

对读请求的马尔可夫空间预测模型的验证如下：不失一般性，我们取磁盘仿真工具 diskSim[4] 使用的两个真实的 I/O 跟踪文件 atlas10k. trace 和 atlas_III. trace 作为测试用例，这两个 trace 是运行 OLTP 的数据库系统的基准测试程序 TPC-C[5] 捕获的 trace，但是它们分布在不同的物理磁盘 Quantum Atlas10K 和 Atlas III 上，即该跟踪文件在不同磁盘上的空间访问特征不一样。我们从这两个 trace 文件中分别提取所有的读操作的起始位置（即磁盘块标号）形成两个序列，即 Atlas10k 的读操作的起始位置序列是：3962268，3962296，2924420，2924276，2000388，8692968，……；Atlas III 的读操作的起始位置序列是：7563004，11589236，11589108，11589040，14486396，……。将上述两个序列分别作为输入参数被我们用 C++ 开发的应用层“读请求的 Markov 空间预测”软件包调用，使用贪婪预测算法作 3-步预测。我们分别重复 1 次、2 次、……、9 次上述序列得到的实验结果如图 3 所示，可以看出随着 I/O 序列的重复次数越来越多，该模型预测的准确性越来越高，重复 1 次预测的准确性约为 48%，重复 9 次预测的准确性约为 87%，并且在两个仿真磁盘上的表现都很一致。

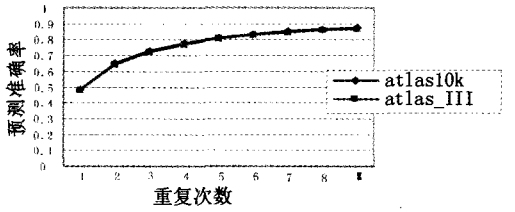


图 3　基于 3-步贪婪预测算法的马尔可夫空间预测模型的准确度

小结　本文设计和实现了一个软件框架——读请求的自动建模器：通过对读请求的空间和时间特性进行综合建模；模型参数被在线、递归地估计；根据新的参数预测将来的读请求。经过实验检验我们的建模和预测比较有效；并且预测的准确性和稳定性都较好。将来的工作是将读请求的空间和时间模型更好地集成，取得令人满意的效果。

参 考 文 献

1　Tran N, Reed D A. Automatic ARIMA Time Series Modeling for Adaptive I/O Prefetching [J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 15(4): 362～377

2　Strang G, Nguyen T. Wavelets and Filter Banks [M]. Wellesley-Cambridge Press, 2002. 102～120

3　I/O traces [EB/OL]. http://www.renci.unc.edu/Project/IO/traces/experimentEntry.asp. 2005～04

4　The DiskSim Simulation Environment. Available at : http://www.pdl.cmu.edu/DiskSim/. 2005, 9

5　Transaction Processing Performance Council. Available at : http://www.tpc.org/. 2005, 9

（上接第 259 页）

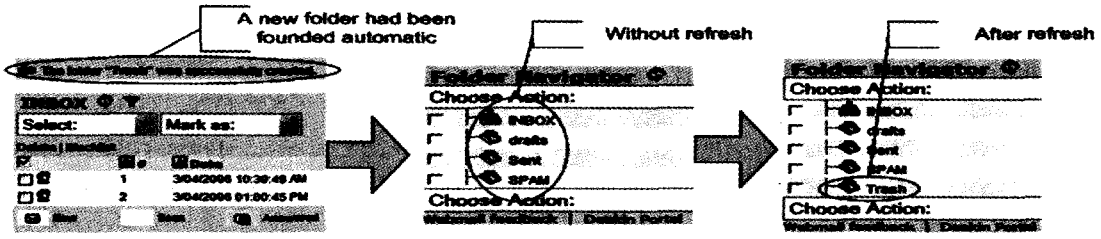


图 9　问题 3 示意图

参 考 文 献

1　Kirakowski J. The Software Usability Measurement Inventory: Background and Usage. In: Usability Evaluation in Industry, Taylor and Francis

2　Bos R, van Veenendaal E P W M. For quality of Multimedia systems: The MultiSpace approach. In: Information Management, May 1998

3　Bass L J, John B E, Kates J. Achieving usability through software architecture: [Technical Report CMU/SEI-2001-TR-005]. Carnegie Mellon University/Software Engineering Institute 2001

4　ISO/IEC FCD 9126-1. Information technology - Software product quality - Part 1 : Quality model. International Organization of Standardization, 2001

5　ISO 9421-10. Ergonomic requirements for office work with visual display terminals (VDT's) -Part 10 : Dialogue principles, International Organization of Standardization, 1994

6　ISO 9241-11. Ergonomic requirements for office work with visual display terminals (VDT's) -Part 11 : Guidance on usability, International Organization of Standardization, 1995

7　Ritter F E, Baxter G D, Jones G, et al. Supporting Cognitive Models as Users 2000 ACM, 1073-0516/00/0600-0141

8　Maloner-Krichmar D, Preece J. A multilevel analysis of sociability, usability, and community dynamics in an online health community. ACM Transactions on Computer-Human Interaction, 2005, 12(2): 201～232

9　Vanderdonckt J, Chieu Chow Kwok, Bouillon L, et al. Interaction Model-based design, generation and evaluation of virtual user interfaces. 2004 ACM 1-58113-845-8/04/0004

10　Vandervelpen C, Coninx K. Towards model-based design support for distributed user interfaces. In: NordiCHI '04, Tampere, Finland, October, 2004

11　Koski T. Hidden Markov Models for Bioinformatics. Kluwer Academic Publishers

12　Lai Xiangwei, Yang juan, Qiu Yuhui, et al. A HMM based Adaptation Model Used in Software Usability Monitoring, ICYCS, 2006

13　Baxter G D. Perspection on CHI. Addison-Wesley, 1999

14　McGregor J D, Sykes D A. A Practical Guide to Testing Object-oriented Software. Addison-Wesley, 2001

15　Bevan N. Quality and usability: a new framework. In: van Veenendaal E, McMullan J, eds. Achieving Software Product Quality, Tutein Nolthenius, 's Hertogenbosch, The Netherlands, 1997