一种基于Markov模型的自适应软件可用性监测模型

赖祥伟 杨 娟 邱玉辉 张为群
(西南师范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 软件可用性工程是当前软件开发的重要研究领域,研究者试图通过对软件可用性特征的分析提高使用者使用效率,改善软件质量。本文针对软件使用过程中的用户操作特征,采用Markov模型对用户软件操作过程进行聚类和建模。并在此基础上根据用户模型的概率描述指标,提出了相应的在线模型更新算法和帮助系统设计方法。在实际软件设计过程中设计实现了以此为理论基础的试验系统,取得了较好的应用效果。

关键词 可用性工程,Markov,HCI,首达率,回访率

A Markov Based Adaptation Model Used in Software Usability Monitoring

LAI Xiang-Wei YANG Juan QIU Yu-Hui ZHANG Wei-Qun
(Faculty of Computer and Information Science, Southwest China Normal University, Chongqing 400715)
(Chongqing Intelligence Software and Software Engineering Lab. Chongqing 400715)

Abstract Software usability engineering is one of the most important research areas in software developing. Researchers can improve the quality of the software through analyzing the characteristics of the software usability. This paper proposes a new Markov based method to cluster and model the users' using processes by collecting the using characteristics of the users during their using processes. We also describe the indices used in refreshing the algorithms corresponded in the online models and designing the helping systems, which is based on the user models' probability. We implement the experiment systems according to this theory proposition during the software design processes, and get an effective reflection either.

Keywords Usability engineering, Markov model, HCI, First accessible ratio, Re-visiting ratio

高可用性的产品可以减少使用者操作错误,减少人员培训和技术支持费用,从而提高产品市场竞争力。规范地说,可用性是指产品为特定使用者用于特定用途时所具有的有效性(effectiveness)、效率(efficiency)和使用者主观满意度(satisfaction)[1]。

1 背景

在影响软件可用性的诸多因素中,人们一直强调"以用户为中心的设计(User-centered design)"。要很好地完成软件可用性设计必须使用到人-计算机交互(HCI),用户界面设计(UID),人类因素学(HF),人类工效学等相关领域的知识[2]。而如何有效地检测用户当前的使用行为和操作状态是所有工作的基础。现有的方法主要包括使用者自我报告、生物特征识别等。

其中,使用者自我报告是一种简单而实用的方法,但其缺点在于是一种事后的方法,同时由于使用者对于可用性测试的目的和方法的不了解导致使用者报告时无法做出有针对性的表述。随着生物特征识别技术的发展,使用者生理特征的检测和判定也成为了可用性测试的一种重要手段,其中美国麻省理工学院(MIT)[3,4]、日本东京科技大学、美国卡内基·梅隆大学等机构在这方面的研究一直处于领先地位。本文的研究主要集中于软件使用者操作过程的检测和建模,并以此为依据改善软件的可用性特征。

2 使用者操作信息的采集

软件使用者在使用软件过程中的操作过程是软件使用者对于软件最为直接的反馈,也是软件可用性评价的重要信息来源。当前常用的使用者操作过程采集方法主要包括查看log文件、操作过程录像、使用专用操作记录工具等。

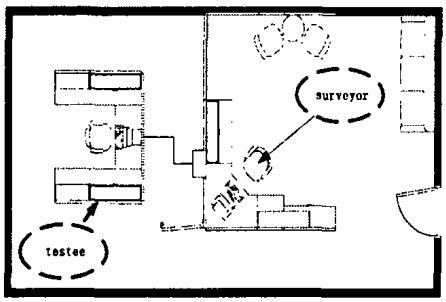

图1 测试环境设计

由于一般的通用软件无法采用类似于Web应用软件那样通过参看log文件的形式来事后采集使用者的操作过程特征数据,因此在试验过程中,我们采用了Mercury Interactive(MI)公司的通用软件测试工具Winrunner来录制使用者的全部操作过程。借用Winrunner高效而准确的使用者操作录制能力,该软件可以完整而清晰地记录软件使用者的全部操作过程。这些过程脚本将在后期处理过程中被详细地分析。

在测试环境的设计过程中,为了尽可能地降低外部环境对于使用者操作过程的影响,测试被设计在一个常规的可用性测试标准环境中进行[5]。测试环境的布置如图1所示。测试用计算机环境如图2所示。

赖祥伟 博士研究生,研究方向为软件工程、智能软件技术等。杨 娟 博士研究生,研究方向为分布式系统,智能计算方法等。邱玉辉 教授,博士生导师,研究方向为人工智能技术。张为群 教授,研究方向为软件工程及人工智能技术等。

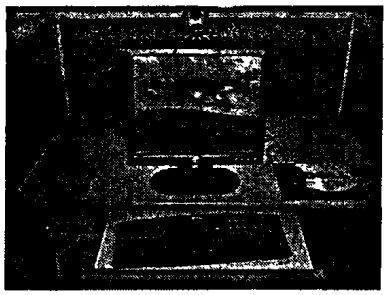

图2 测试用计算机

3 使用者操作过程的 Markov 模型建模

3.1 软件使用者模式聚类

在试验过程中，要求被试者在使用结束后对软件使用情况做出自我报告，报告的内容主要是对软件的总体评价(方便、较为方便、没有特殊体会、稍有阻碍、操作困难)。评价的归一化公式为：

$$R=r/5 \quad r=\begin{cases}5(good)\\4(getter)\\3(normal)\\2(worse)\\1(difficult)\end{cases}$$

这些评价与使用者完成制定功能的平均时间，以及最终工作结果评价三者的加权平均共同构成了软件操作效果的分类。权值的计算公式为：

$$S_i=\eta_1 R_i+\eta_2\frac{T_n}{\prod_{j=0}^{n}T_j}+\eta_3 A\cdot(\eta_1+\eta_2+\eta_3=1)\wedge(A=\begin{cases}0(default)\\1(succeed)\end{cases}) \tag{1}$$

η_1,η_2,η_3是一系列人工设定的参数。在实际使用过程中的经验取值为(0.4,0.2,0.4)。由式(1)的计算结果可以将所有使用者划分为多种不同的层次类型。具体分割函数为：

$$M=\begin{cases}1(S_i\geqslant 0.75)\\2(0.5\leqslant S_i<0.75)\\3(0.25\leqslant S_i<0.5)\\4(S_i<0.25)\end{cases}$$

每个类型的多个使用者的操作序列体现了该类使用者的许多共同的操作特征和使用习惯。由此可以使用其操作过程脚本为每个类型的使用者建立相应的 Markov 过程模型[6]。每个模型中的具体参数是由其属于该模型的多个使用者的操作过程记录通过如下的方法提取得到的。

3.2 使用者操作过程的 Markov 模型

使用者的操作过程可以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)。这个随机过程可以被定义为：

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

在每个操作过程中，使用者为了实现某个既定的操作目标会进行多步的操作。其中有的操作是完成目标所必须的和关键的，同时也存在部分无用的误操作或是一些冗余操作。为了准确地刻画出使用者的操作特征，首要的工作是需要确定操作过程中的关键过程，尽可能去掉冗余操作的影响。判定某过程是否为关键过程需要使用多方面因素进行考量。其中主要的因素包括：

(1)使用者对该状态的访问次数 V_i；

(2)每次访问的持续时间 O_i；

(3)人为设定的权重 m。

在关键过程的判定过程中，单因素的判定是不可靠的，在实际计算过程中，我们综合了上述三方面的因素，采用下列公式来确定某状态的权重。

$$w_n=\left(\frac{v_{in}}{\sum_{i=0}^{n}v_{ni}}+\frac{O_{in}}{\sum_{j=0}^{n}O_{ij}}+\frac{m_n}{\sum_{k=0}^{n}m_k}\right) \tag{2}$$

通过计算操作过程中每个状态的权重，可以按照实际操作目标的复杂程度选择权值最大的 n 个状态作为 Markov 链的转换状态。对应 Markov 模型的转移矩阵为：

$$P_{ij}(n,m)=P(\xi_m=j|\xi_n=i)(m\geqslant n) \tag{3}$$

其中具体的转移概率可以通过对属于该类的所有操作序列的统计得到。

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的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被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。

4 操作过程监控模型

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：

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，指定状态的首达率被定义为：

$$f_{ij}^{(n)}=P(\xi_n=j,\xi\neq j,k=1,2,\cdots n-1|\xi_0=i)(n\geqslant 1) \tag{4}$$

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，可以应认为使用者在完成该功能过程中遇到了一定的困难，这时的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使用情绪的影响。

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 $N_i^{(t)}$，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_n，如果$|S_i-S_n|\leqslant 2e$(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:

$$f(n)=1\Leftrightarrow\exists\ l\cdot P_{ij}(n,n)=\sum_{k}p_{ik}(n,l)p_{kj}(l,n) \qquad (5)$$

公式表明如果使用者在回访某状态前曾经访问过利用式(5)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处于困境。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 核心路径指标

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。

在软件实现过程中为了提高软件可用性特征,可以综合以上方法对软件进行改造,增加软件的自适应能力[8~10]。其中最主要的方法是为系统增加一个在线的帮助系统,该系统的主要功能包括:

(1)监控并记录软件的使用过程;

(2)在线更新使用者模型;

(3)实时计算使用者使用过程中的相关 Markov 参数;

(4)为使用者提供自适应的在线帮助。

系统的实现模型如图3所示。

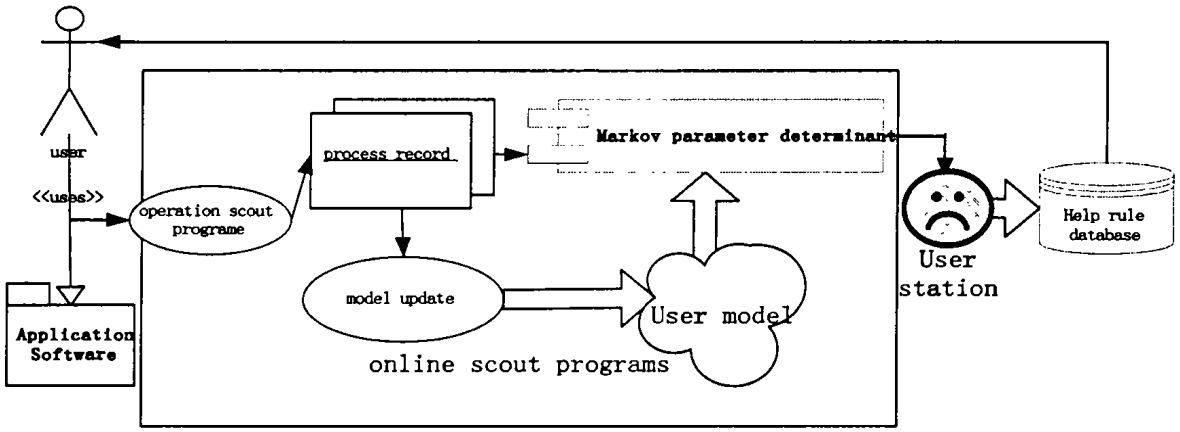

图3 软件使用过程在线监控模型

5 可用性测试实践

为了验证本方法的有效性及效率,我们以一个常用的教学软件进行相关的实际试验。

5.1 被试的选择

为了测试的准确性,我们选择了40名学生作为软件使用者对软件进行使用。其中包括20名计算机专业学生和20名非计算机专业学生,在计算机专业学生中包括10名刚完成1学期学习的新生。

5.2 试验参数的设定

在使用前,所有被试都未被告知测试的目的。只是知道这是一个为了检查软件本身问题的小试验。被试被要求在规定时间内尽量完成同样的学习内容要求。在试验过程中,使用者被要求完成某指定知识的学习任务。时间为20分钟,在学习完成后进行实际考核以验证学生的学习效果。

5.3 试验结果分析

在所有被试中,使用前述的方法对操作过程进行聚类。所有被试的分类的情况见表1。

表1 使用者聚类结果表

分类	最高权值	最低权值	平均权值	人数
1	0.943	0.756	0.843	17
2	0.736	0.507	0.693	7
3	0.482	0.284	0.354	11
4	0.231	0.165	0.224	5
合计	0.943	0.165	0.632	

5.4 增加使用过程监控系统后的对比试验数据

在上述试验数据的基础上,我们根据软件测试过程中所得到的相关数据对软件的操作过程和顺序进行了相应的调整和改进,并增加了相应 primary key 的附加帮助系统。

改进后采用上述方法对软件的使用者操作进行了对比试验,结果表明,使用者操作的聚类权值从0.632增加到0.717,使用者自我报告满意度从42.5%上升到57.5%。软件可用性有了明显的提高。

结论 通过对相关理论和试验的总结可以发现,Markov 模型对于软件使用者的操作过程有着很强的描述能力和预测能力。通过对于聚类参数的调整,可以针对实际情况建立更精确的模型,从而增加系统对于复杂系统的适应能力。但系统对于缺乏关键过程软件的建模能力需要进一步验证。

参 考 文 献

1 ISO 9241-11 Ergonomic requirements for office work with visual display terminals (VDTs)-Part 11: Guidance on usability
2 Nielsen J. Structured Usability Engineering Elsevier Inc. 1994
3 Mota S, Picard R W. Automated Posture Analysis for Detecting Learner's Interest Level, MIT, 2003
4 Picard P W, Klein J. Computers that Recognize and Respond to User Emotion: Theoretical and Practical Implications: [MIT Media Lab Tech Report 538]. 2002
5 Rabiner L R. A Tutorial on Hidden Markov Model and Selected Application in Speech Recognition. IEEE77(2): 257~285
6 龚光鲁,钱敏平. 应用随机过程教程及其在算法和智能计算中的随机模型. 清华大学出版社,2004
7 Johnson P H, Designers-identified requirements for tools to sup-

porttask analyses. In. Proc. of INTERACT'90, 1990. 259～264
8 Johnson H P. Task knowledge structures: psychological basis andintegration into system design. Acta Psychologica, 1991, 78: 3～26
9 Lim K Y, Long J B. Instantiation of Task Analysis in a StructuredMethod for User Interface Design. In: Proc. of Task Analysis in H-C-I, 11th Interdisciplinary Workshop on "Informatics and Psychology", Schaerding, Austria, June, 1992
10 Dumas J S, Redish J C. A practical guide to usability testing. Ablex Publishing Corporation, Norwood, 1994

(上接第208页)

分析当前的性能数据，可以得到系统运行到该节点的CPU执行的总周期数，出口与入口两者的差就是该层运行所消耗的CPU时钟周期数，也就是该层的时间性能。考察图2，为了得到线程1各个关键边的执行代价，我们在顶点 V_{13}, V_{14}, V_{15}, V_{16}, V_{17} 处分别插入PAPI CPU时钟周期数 *PAPI_TOT_CYC* 采集函数 *PAPI_read* (*EventSet*, *values*)，其中EventSet为当前所监控的事件集{*PAPI_TOT_CYC*}，values返回监控的事件值。之后，运行系统，就可以得到所有监控点的 *PAPI_TOT_CYC* 值，然后通过计算分析，得到表1第3行的各个线程的性能数据，关于PAPI的详细描述可参阅文[11]。

6 实验结果与分析

在图1中标明了通过实验得到的运行时各个线程的动态时钟周期数，然后按照第5节给出的方法，在第2步的时候我们得到图2(a)的结果，就是在简化第一层后的简化的同步图。然后进一步简化得到图2(b)，这样就得到最后的系统性能数据表1。

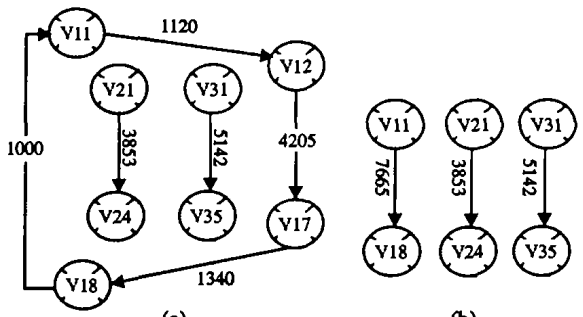

图2 同步简化图

从图1的边的权重值可以看到，所有的信号量消费者的位置，其时钟周期数在1700左右，而在生产者的位置，为400左右，这个差值很明显就是两个线程间的同步代价，因为对于信号量的消费者来说，它需要等待信号量生产者发出信号，而对于生产者来说，直接赋值即可产生信号量(此处忽略同步临界区进入时间，因为无论对于生产者还是消费者，都有这样一个消耗)，可以看出对于一个并发多线程的系统，同步通信代价是不可随意忽略的，而采用PERC-PAPI技术得到的性能数据和分析结果有比较高的精确度。

表1 在不同分析技术下的各个进程的性能数据比对

类型	P1	P2	P3	系统	误差比
期望值(E)	7300	3600	4800	7300	0%
静态分析数据(S) 忽略同步代价	6500	2800	4100	6500	−11.43% ((S-E)/E)
采用PAPI动 态采集(P)	7665	3853	5142	7665	+4.79% ((P-E/E)

结论 本文引入了一种动态性能分析技术，并将其应用到了并发线程通信系统的性能评估中。通过本文的论述，我们可以看到在实际应用中并发线程间的同步通信代价对系统的整体性能影响是不可随意忽略的，通过引入PERC-PAPI技术，使得对于这一类系统的性能分析和评估的精确度得到较大提高。在今后的研究中，将考虑应用这种技术到集群并行处理应用中，同时因为现在的分析是完全依赖于源代码级的"探针"插入方式得到，使其应用有一定的局限性，因此也要考虑如何完成不需要源代码而直接插入和监控目标代码进行分析。通过PAPI监控技术实际可以得到更多的除了时间性能之外的程序行为的底层表现，如Cache(高速缓存)和TLB(Tranlate Lookaside Buffer)的使用情况，分支预测情况等，从而可以提供给系统分析和结构设计师更多和更小粒度的性能数据，进一步优化和完善应用系统，特别是并行系统和多线程并发系统，也可用于这类系统的运行状态监控中。

参考文献

1 Bhattacharya S, Dey S, Brglez F. Performance Analysis and Optimization of Schedules for Conditional and Loop-Intensive Specifications. In: Proc. Design Automation Conf. June 1994. 491～496
2 Dey S, Bommu S. Performance analysis of a system of communicating system. C&C Research Laboratgories, IEEE, 1997
3 Zhang Tianhao, Cao Feng, Dewey A M, et al. Performance Analysis of Microelectrofluidic Systems Using Hierarchical Modeling and Simulation. IEEE, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, 2001, 48(5)
4 Wilkinson B, Allen M. 陆鑫达译. 并行程序设计. 机械工业出版社, 2002. 50～62
5 Kuehlmann A, Bergamaschi R A. Timing Analysis in High-Level Synthesis. In: Proc. of the IEEE Intl. Conf. on computer-Aided Design, Aug. 1992
6 Narayan S, Gajski D D. System clock Estimation based on Clock Slack Minimization. In: Proc. of the European Design Automation Conf. 1992
7 London K, Dongarra J, Moore S, Mucci P, Seymour K, Spencer T. End-user Tools for Application Performance Analysis, Using Hardware Counters. In: Intl. Conf. on Parallel and Distributed Computing Systems, Dallas, TX, Aug. 2001
8 Brewer F, Gajski D. Chippe: A System for Constraint Driven Behavioral Synthesis. IEEE Transactions on Computer Aided Design, 1990, 9: 681～695
9 Miller J, Kramer H. Analysis of multi-process specifications with a petri-net model. In: Proc. EURODAC'93, European Design Automation conference with EURO-VHDL'93, 1993. 474～479
10 Bailey D H, Supinski B D, et al. Performance Technologies for Peta-Scale Systems: A White Paper Prepared by the Performance Evaluation Research Center and Collaborators. Response to the Invitation to Submit White Papers on High End Computing, May 2003
11 ICL of University of Tennessee. PAPI Programmer's Reference. http://icl.cs.utk.edu/