基于UMMs的Web软件统计测试方法研究

黄 娟 张为群 闻 晓 梁智远

(西南大学计算机与信息科学学院 重庆 400715)

摘 要 随着网络使用的普及以及信息技术的不断进步,如何保证Web软件的可靠性显得越来越重要。统计测试和可靠性分析能有效地确保Web软件的质量。利用从Web日志中提取的访问信息和错误信息,提出一种基于统一的马尔可夫模型(UMMs)的Web软件统计测试方法,该方法将访问信息用于构建UMMs,相关的错误信息用于评价Web软件的可靠性和Web统计测试的有效性。最后将该方法用于分析某Web软件,以实验证明该方法的可行性和有效性。

关键词 Web软件,统计测试,可靠性分析,统一的马尔可夫模型

中图法分类号 TP311 **文献标识码** A

Research for UMMs-based Statistical Testing Method of Web Applications

HUANG Juan ZHANG Wei-qun WEN Xiao LIANG Zhi-yuan

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

Abstract With the prevalence of Web applications and the development of technology, it brings concerns related to the reliability of Web application. Statistical testing and reliability analysis can be used effectively to assure quality for Web applications. Using the Web usage and failure information from existing Web logs, this paper presented a statistical testing method based on UMMs. The usage information was used to build UMMs. The related failure information was used to measure the reliability of Web applications and the potential effectiveness of statistical Web testing. At last we applied this approach to analyze one Web application. The results demonstrated the viability and effectiveness of our approach.

Keywords Web application, Statistical testing, Reliability analysis, Unified markov model

1 引言

Web应用为众多用户提供了跨平台的统一访问。随着互联网和Web应用的普及,如何确保Web应用的可靠性显得越来越重要。目前虽然存在各种各样的测试工具和方法,如:功能测试、性能测试,但是由于Web应用的特点是体系结构复杂、代码量大、页面众多且相互连接,如果基于传统的覆盖测试,在一定的时间和投入内,显然无法做到穷尽的测试。事实上软件应用的某些部分的执行频率远远高于其他部分[1]。统计测试正是基于这一事实,标示出那些频繁执行的部分,并相应地调整测试策略,针对这些频繁执行的部分进行详尽的测试。通过提高关键模块的安全性和可靠性,来提高整个系统的安全性和可靠性,以提高测试的性价比。因此,用于确保Web应用可靠性的一个很好的可选方案就是统计测试并进行相关的可靠性分析。

统计测试进行的前提条件就是生成如实反映系统使用情况的使用模型,以指导测试的进行。以往使用模型的建立主要是通过预测和估计或者对终端用户的调查建立的,不能如实地反映系统的真实情况。对Web应用而言,用户的使用信息和系统的错误信息作为日志文件被保存在服务器端。我们可以通过从日志文件中提取使用信息来得出系统各部分的使用频度,分析失效信息,进行可靠性分析。

2 统计测试的必要性

Web应用软件一般都具有体系结构复杂、代码量大、页面众多且相互联结的特点,在一定的时间和投入内,显然是无法做到穷尽的测试,而确保其可靠性的一个很好的可选方案就是基于使用模型的统计测试。它使我们将关注点放在频繁使用的部分和特征上,从而有效地确保了软件的可靠性,它主要分成以下3个部分[2]。

(1)根据使用情景和相关的使用频率构建使用模型。

(2)根据使用模型进行测试案例的构建、选取和执行。

(3)对测试结果进行分析,以进行可靠性的评估和预测。

3 UMMs的提出及形式化定义

Web统计测试的第一个步骤就是根据使用场景构建模型,然后基于使用模型选择和设计测试用例,最后分析测试结果,评估应用软件的可靠性。因此,使用模型非常重要,它表

到稿日期:2008-11-10 本文受到重庆市自然科学基金重点项目"软件测试技术和方法研究"(CSTC,2006BA2003)资助。

黄 娟(1983—),女,硕士研究生,研究方向为软件工程、Web软件测试,E-mail:juanjuan662001@163.com;张为群(1950—),男,教授,硕士生导师,计算机学会会员,主要研究领域为形式化软件工程、软件测试;闻 晓(1980—),男,硕士研究生,研究方向为软件工程、Web软件测试;梁智远(1983—),男,硕士研究生,研究方向为软件工程、形式语言。

示了一个 Web 应用软件的使用操作(指在预期的操作环境中的预期使用)。

文献[9]提出一种扁平操作框架的使用模型,这种模型有些简单,不能满足 Web 软件不断复杂化的需要。文献[10]提出一种基于马尔可夫的使用模型,这种模型在一个马尔可夫链中显示了通常使用的操作单元,而状态转移的概率与历史信息无关,由相关人员给出。完整的操作或者导航模式可以通过将不同的状态利用转移连接到一起来构建,整个路径的概率由每一个转移的概率求乘积来得到。由于 Web 软件比较复杂,这种基于马尔可夫链的使用模型比较适合 Web 软件的统计测试,但是这种模型只是覆盖了用户使用情况的某些方面,对于一些复杂的使用情况难以建模。

在此基础上 J. Tian 等人[11,12]提出一种 UMMs(Unified Markov Models)的使用模型,提出一种层次化的方法进行建模。UMMs 可以捕获执行流、信息流、事务处理和相关的使用信息,这些信息通过一系列关联的、层次化的 Markov 来表示,这种层次化的结构具有很好的灵活性。本文将 UMMs 用于 Web 软件的统计测试中,基于 Web 应用的导航行为图和日志文件进行构建,模型图中包含了导航连结、动态构建条件、转移标识和转移概率等信息,对用户的行为描述更为全面,能够满足复杂 Web 应用的统计测试。

UMMs 的基本组成要素包括:状态节点、状态转换和状态间的转换概率,将 UMMs 定义为六元组[3]:$M=\langle S,\Gamma,\delta,s_0,f,n\rangle$,其中:

S 是软件使用的状态集合,软件的每个状态定义为 $s\in S$ 为一个二元组$\langle Statenum, Notation\rangle$。其中,$Statenum$ 是状态的编号,$Notation$ 是该状态的附加信息[4]。

Γ 是转换标记的集合,状态转换标记通常有转换输入和转换概率构成,每个转移标记定义为 $\Gamma=\langle labelName, pf\rangle$。其中,$labelName$ 是转移标记的名称,pf 是该转移发生的概率。

δ 表示软件执行过程中软件状态之间的转移函数,$\delta: S\times\Gamma\rightarrow S$。

$s_0\in S$,是初始状态,它代表软件激活的状态。

$f\in S$,是终止状态,它代表软件终止执行时进入的状态。

n 是模型的层次。

4 Web 软件统计测试方法

4.1 构建 UMMs

本文提出的 Web 软件统计测试的使用模型 WSUM 利用 Markov 链来进行构建,主要表示了 Web 软件的序列、执行的过程中所提供的输入值和每一次转移的概率。利用页面导航图,将每一个页面看作一个状态,页面中导航看作 Markov 链的边,然后将其进行适当的转换,其次通过分析日志得到的每个转移的概率来修饰转换后的图,最后将关联的模型之间进行层次化,得到 Web 软件使用模型 UMMs。具体过程如图 1 所示。

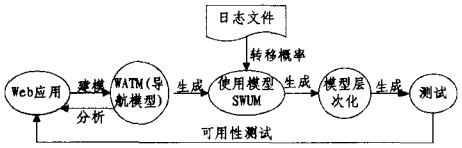

图 1 UMMs 生成过程

构建能够描述一个用例所有执行过程和执行概率 UMMs 的算法如下:

算法 1 构造单个用例的状态转换模型作为单个用例的使用模型[5]:

Step1 构造一个 Markov 模型

1. 创建一个 Markov Chain,S_0 是初始状态,Pf←场景发生概率,当前状态为 S_0

2. For 所有消息 m,

(1)构造迁移 t,标记为〈Transition,m,pf〉

(2)增加新的状态 S,ADD (当前状态,S,t),S←场景结束时进入的状态

EndFor

Step2 集成所有其他的顺序图,构造用例的 Markov 模型

1. For 所有其它场景,当前状态←S_0

2. For 该场景的所有消息 m

(1)构造迁移 t,标记为〈Transition,m,pf〉

(2) If t 在已生成模型中已经出现过

Then 累加其在生成模型中的迁移率

Else //增加一个新状态

ADD (当前状态,S,t)

End If

EndFor

3. If 本场景结束时的状态已存在于模型中

Then 不把该状态加入模型中

Else

当前状态本场景结束时的状态

End If

EndFor

Step3 对各个状态的出迁移的迁移概率进行范化,使其和为 1。

算法 2 集成用例 Markov 链构造软件的使用模型 UMMs。根据之前得到的用例之间的执行顺序关系,用每个用例的前置条件以及用例各场景执行后的状态来构建。

```
For(i = 1,i < n,i ++) / * 模型的层数 * /
 For (j = 1,j < n,j ++)
  For 每个与该用例相连的用例模型
   If Sj. out = S0. in
    Then 合并 Sj. out 和 S0. in
   End If
  EndFor
 EndFor
EndFor
```

图 2[1]是 UMMs 的一个例子,图 2(a)表示了最高级别的操作单元(状态),相关的转移旁边具有转移发生的概率。其中的一些状态可以更为详细地进行建模,例如图 2(b)扩展了 Programs 状态,图 2(c)扩展了 Masters 状态,这些不同细节级别的 Markov 链集合形成了一个层次结构,对于 Web 统计测试非常重要。

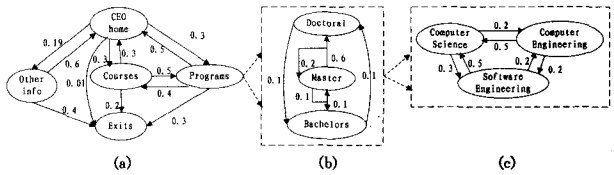

图 2 UMMs(Unified Markov Models) for SMU/CSE Web pages

4.2 由 UMMs 生成测试用例

(1)对需要测试的路径设定一个门限值。测试用例可以选取使用模型中状态节点和状态节点之间的连接的路径,通

过选择超过门限值的测试用例来完成覆盖使用频率较高的操作，以此提高软件的可靠性。

(2)调整门限值，控制需要生成和执行的测试用例的数量。例如，可以开始的时候设定较高的门限值，这样只对频繁进行的操作进行测试，然后逐渐降低测试的门限值，对更广泛的状态进行测试，以达到更高的执行满意度和软件可靠性。对用户的操作采用总体的门限值来确保终端用户进行的操作被覆盖并被充分地测试[7]，核心执行算法如下：

```
TestProgram(UsecasePath)
{
  Initialthreshold1=X; /*对需要测试的路径设定一个初始门限值*/
  IF threshold(usecase)>X; /*被选测试用例的门限值大于设定的门限值*/
    THEN excute(usecase); /*执行该测试用例*/
  End If
  While(IsProgram) /*在该门限值的设定下系统测试是否完成*/
  {
    Initialthreshold2=Y; /*降低测试的门限值,Y<X*/
    IF threshold(usecase)>Y; /*被选测试用例的门限值大于再次设定的门限值*/
      THEN excute(usecase); /*执行该测试用例*/
    End If
  }
}
```

例如：在图 1 中设定测试序列的门限值为 0.02，对于测试用例“CSE home->Courses ->Exit”，这条执行路径的转换概率：0.3*0.2=0.06，达到了测试的门限值，所以这个相应的测试案例被选定并被执行。但是，对于测试用例“CSE home ->Exit”，这条执行路径的转换概率为 0.01，没有达到测试的门限值，所以这个相应的测试用例没有被选中。

4.3 软件可靠性评估

通过对日志的分析，我们可以对错误出现的分布和软件可靠性做出评估，很多自动化软件对此提供了支持，如：“the log analyzer's Analog，FastStats”。合适的选取使用和错误数据，能够通过各种合适的软件可靠性模型，来提供一个当前 Web 应用可靠性的快照。例如，如果 n 次的用户访问中总的错误数被记录，则根据 Nelson 模型[8]，软件的可靠性被计算得出：$R=1-\frac{f}{n}$（f 为错误数）

如果每次访问的时间能够被记录，则平均失效可以被计算得出：$MTBF=\frac{1}{f}\sum_i t_i$

如果不能有效地评估每次访问的时间，则可以使用访问的次数作为参数进行计算。在这种情况下，平均失效可以被计算得出：$MTBF=\frac{n}{f}$

5 在 SWU/CS Web 软件中应用统计测试方法

用于该实验的 Web 软件为 University of SouthWest's Computer Science(SWU/CS)。根据时间的不同，我们进行了两次实验。实验 1 中采用的服务器访问日志和错误日志信息来自大学假期之间，实验 2 中采用的服务器访问日志和错误日志信息来自学生在校学习期间，两个实验的时间设置都为 26 天[1]，我们认为不同的时间设置可以代表不同的使用模式，同时我们也想证明多重的 UMMs 设置对一个 Web 软件是否有必要。

5.1 对 SWU/CS Web 软件构建 UMMs

我们对实验 1 中 SWU/CS 访问日志的分析使用日志分析工具 FastStats，FastStats 将搜集的数据和存储的相关信息转换成报表、图表和报告，得到的连接树图[6]的统计信息构建相应的 UMMs 图，如图 3 所示。

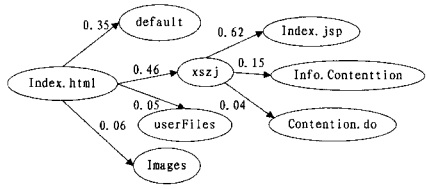

图 3 构建的 UMMs

5.2 评价 SWU/CS Web 软件的可靠性和 UMMs 的有效性

由于 FastStats 对错误日志分析的结果不能满足我们的需要，因此我们需要从错误日志中提取更多详细的错误信息。和大多数常见的错误类型相比，“文件不存在”和“访问受限”是两类最常见的错误，而且前者错误数量远远超过后者。对实验 1 和实验 2 进行统计分析后得到的数据如表 1 所列。

表 1 两次实验数据

	f(错误数)	n(访问量)
实验 1	13212	878323
实验 2	28336	1755152

采用 Nelson 模型和 MTBF，实验 1 对该 Web 软件的可靠性评估计算结果为：

Reliability=1−13212/878323=98.50%

MTBF=878323/13212=66.48

实验 2 对该 Web 软件的可靠性评估计算结果为：

Reliability=1−28336/1755152=98.39%

MTBF=1755152/28336=61.94

得到的错误数和访问量相关的图形如图 4 和图 5 所示。

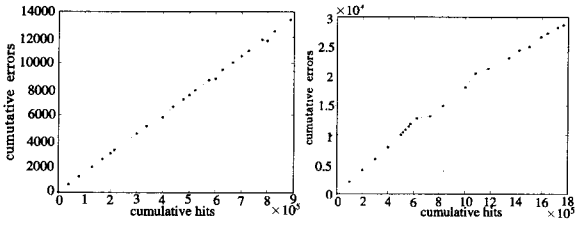

图 4 错误数 V 访问量(实验 1) 图 5 错误数 V 访问量(实验 2)

从图中我们可以看到使用频度和错误的发生率之间呈线性关系，也就是说错误的发生数与 Web 软件的使用量按比例增加。比较实验数据，我们可以看到实验 2 的 Reliability 和 MTBF 值比实验 1 低一点，但是两个实验的值近似，因此我们判定 SWU/CS Web 软件是稳定的，并且是可靠的。

5.3 对比实验

对该 Web 软件采用文献[7]中提出的基于马尔可夫链使用模型的统计测试方法，由于 Web 比较复杂，这种模型只是覆盖了用户使用情况的某些方面，对于一些复杂的使用情况难以建模，测试得到的数据如表 2 所列。

表 2　两次实验数据

	f(错误数)	n(访问量)
实验 1	10230	267801
实验 2	20153	1755152

采用 Nelson 模型，实验 1 和实验 2 对该 Web 软件的可靠性评估计算结果分别为：

Reliability1＝1－13212/878323＝96.18%

Reliability2＝1－28336/1755152＝95.47%

将两种统计方法得到的 Web 软件可靠性(R)对比，结果如表 3 所示。

表 3　实验数据对比

	实验 1	实验 2
马尔可夫链使用模型	96.18%	95.47%
UMMs	98.50%	98.39%

通过实验数据对比我们可以发现，采用本文提出的基于 UMMs 的统计测试方法能够对 Web 软件进行更详尽的测试，提高测试的覆盖率和软件的可靠性，能够满足复杂 Web 应用软件的统计测试。

结束语　本文将 Web 软件统计测试方法应用于 SWU/CS Web 软件，分析该软件的 Web 日志信息，构建该 Web 软件的 UMMs。为了评估该软件的可靠性，我们将统计得出的访问数据和错误数据用于 Nelson 模型，计算得出该软件的可靠性值和 MTBF，以此证明该软件是稳定的和可靠的。同时也说明 UMMs 对于 Web 软件的统计测试使用模型是合适的，多重的 UMMs 构建对于判断一个 Web 软件的可靠性更有必要。最后将该方法和基于马尔可夫链模型的统计测试方法进行对比，证明该方法能够提高测试的覆盖率和软件的可靠性。

参 考 文 献

[1] Kallepalli C, Tian J. Measuring and modeling usage and reliability for statistical Web testing. IEEE Trans. Software Eng., 2001, 27(11): 1023-1036

[2] 沈宏. 基于用例和日志的 Web 统计测试. 硕士学位论文. 上海师范大学, 2004

[3] Tian J, Ma L, Li Z, et al. A hierarchical strategy for testing Web-based applications and ensuring their reliability // Proc. of the 27th Annual International Computer Software and Applications Conference. 2003

[4] 张慧，熊前兴. 基于 UML 的软件统计测试方法研究[J]. 计算机与数字工程, 2008, 36(1): 22-22, 25

[5] 邓晔，徐锡山，颜炯. 基于 UML 的软件使用模型的研究及工具实[J]. 计算机应用研究, 2006, 23(1): 129-131

[6] Hao Jian-hua, Emilia M. Usage-based Statistical Testing of Web Applications. ICWE'06, Palo Alto, California, USA, July 2006

[7] 路晓丽. Web 应用软件的测试技术研究. 西北大学博士学位论文. 2006

[8] Walton G H, Poore J H. Statistical testing of software based on a usage model. Software-practice and Experience, 1995, 25: 97-108

[9] Karlin S, Taylor H M. A First Course in Stochastic Processes, second ed. New York: Academic Press, 1975

[10] Whittaker J A, Thomason M G. A Markov Chain Model for Statistical Software Testing. IEEE Trans. Software Eng., 1994, 20(10): 812-824

[11] Tian J, Lin E. Unified Markov for Models Software Testing, Performance Evaluation, and Reliability Analysis // Proc. Fourth ISSAT Int'l Conf. Reliability and Quality in Design. Aug. 1998

[12] Tian J, Nguyen A. Statistical Web Testing and Reliability Analysis // Proc. Ninth Int'l Conf. Software Quality. Oct. 1999: 263-274

(上接第 272 页)

从测试结果可以看出，GobackFS 文件系统和 CopyFS 文件系统的性能基本相同，在读写方面都略低于 Ext3 文件系统，但在寻道操作方面均优于 Ext3，这可能和 FUSE 框架的底层实现有关。

结束语　我们把一个进程在其执行过程中对文件的一系列修改称为一个事件。为了解决如何方便有效地恢复某个事件对文件的修改，我们研究并实现了一个具有事件恢复功能的文件系统 GobackFS。GobackFS 文件系统基于 FUSE 框架实现，是一个用户空间的文件系统，可以方便地 mount 到通用 Unix/Linux 文件系统上使用。测试结果表明，GobackFS 文件系统的读写性能略低于 Ext3 文件系统，但寻道速度略高于后者。

目前，GobackFS 文件系统提供的功能还比较简单，我们下一步的工作主要包括两个方面：第一，为用户提供差分备份的功能和磁盘空间的控制功能。第二，用户可以选择性地备份文件，比如用户可以选择不备份某些格式的文件，如 mp3 和 avi 格式等。

参 考 文 献

[1] Kistler J J, Satyanarayanan M. Disconnected operation in the Coda file system // Proceedings of 13th ACM Symposium on Operating Systems Principles. Asilomar Conference Center, Pacific Grove, CA, ACM Press, October 1991: 213-225

[2] Lee E K, Thekkath C A. Petal: Distributed virtual disks // Proc. of the 7th Conference on Architectural Support for Programming Languages and Operating Systems. 1996

[3] Chervenak A, Vellanki V, Kurmas Z. Protecting file systems: A survey of backup techniques // Proceedings of the Joint NASA and IEEE Mass Storage Conference. 1998

[4] Azagury A, Factor M E, Satran J. Point-in-time copy: Yesterday, today and tomorrow // Proceedings of the Tenth Goddard Conference on Mass Storage Systems and Technologies. 2002: 259-270

[5] Gifford D K, Needham R M, Schroeder M D. The Cedar File System. Communications of the ACM, 1988, 31(3): 288-298

[6] Santry D S, Feeley M J, Hutchinson N C, et al. Deciding When to Forget in the Elephant File System // Proceedings of the 17th ACM Symposium on Operating Systems Principles. December 1999: 110-123

[7] Muniswamy-Reddy, et al. A Versatile and User-Oriented Versioning File System // Proc. of the 3rd USENIX Conference on File Storage and Technologies. March 2004

[8] Zadok E, Nieh J. FiST: A Language for Stackable File Systems // Proceedings of the Annual USENIX Technical Conference. June 2000: 55-70

[9] Cornell B, Dinda P A, Bustamante F E. Wayback: A user-level versioning file system for Linux // Proceedings of the USENIX Annual Technical Conference. FREENIX Track. 2004: 19-28

[10] Szeredi M. Filesystem in USEr space. http://sourceforge.net/projects/avf, 2003

[11] Peterson Z, Burns R. Ext3cow: A Time-Shifting File System for Regulatory Compliance. ACM Transactions on Storage, 2005, 1(2): 190-212

[12] http://sourceforge.net/projects/lufs/

[13] Vigier N, Joubert T. http://n0x.org/copyfs/

[14] Bray T. The Bonnie Benchmark. http://www.textuality.com/bonnie