# 用 Monte Carlo 模拟方法评估 Web Services 系统的可靠性

支小莉，陆鑫达
（上海交通大学计算机系，上海 200030）

**摘 要：** 现有文献很少考虑到 Web Services 系统（WSS）这类具有随机并行度的多用户系统的可靠性问题．本文在为 WSS 建立一个基于体系结构的随机服务系统模型的基础上，提出一种模拟的解决方法及其实现系统 Simurel．通过引入运行概图和失效概图，Simurel 取得了良好的可应用性，可以用于一般性的基于体系结构的可靠性评估，不论是多用户或单用户系统．实验结果表明，Simurel 的运行性能良好．

**关键词：** Monte Carlo 模拟；多用户；软件可靠性；Simurel
**中图分类号：** TP311.5 **文献标识码：** A **文章编号：** 0372-2112 (2002) 12A-2172-05

# Using Monte Carlo Simulation to Evaluate the Reliability of Web Services Systems

ZHI Xiao-li, LU Xin-da
(*Dept. of Computer Science, Shanghai Jiaotong Univ., Shanghai* 200030, *China*)

**Abstract:** The current research work considered nothing explicitly about the reliability problem for the multi-user system of random degree of parallelism, such as the Web Services system (WSS). By modeling WSS as architectural random server system, this paper proposes a simulation-based approach and its implementation system, Simurel. Making use of execution profiles and failure profiles, Simurel reveals good applicability and generalization. It can be used to solve architecture-based reliability assessment in a general way, no matter single-or multi-user. The results of several experiments indicate that Simurel bears good runtime performance.

**Key words:** Monte Carlo simulation; web services system; software reliability; simurel

## 1 引言

随着现代社会对软件系统的依赖程度越来越大，软件可靠性作为软件质量度量的一个重要指标得到了研究者和业界的广泛重视．Web Services 系统（简写为 WSS）是由自治的 Web Service 松耦合的系统．WSS 被认为是下一代 Internet 计算的主流模式，并且是微软的．NET 战略、Sun One 平台和 W3C 的 XML 活动的核心[1]．WSS 由于具有独特的性质：大用户数，部件自治等，使得它的可靠性评估出现新的问题．

评估软件可靠性一般有两种方法：分析模型方法[1,4]和随机模拟方法[6]．由于分析模型的过份理想的模型假设和严格的数学精确性使得它们的实际应用受到较大的限制．而模拟方法通过模拟系统行为而不是采用精确的数学公式克服了这些缺陷而成为一种具有前途的可靠性评估技术．签于用户到达的随机性及动态并行度，用分析模型求解 WSS 的可靠性是非常困难的，模拟方法应是一个可行方案．

另外，现有的可靠性方法按研究角度不同大致可归类为黑盒法与白盒法．黑盒法从系统的角度研究软件的可靠性增长模型，文献[1]是这方面的一本很齐全的参考书．白盒法即基于体系结构（architecture-based）的方法，根据模块间的结构信息与单个模块的可靠性来评估整个系统的可靠性．这方面的综述请参见[4]．由于 WSS 是多个自治 Service 的松耦合，基于体系结构的方法能提供一种自然的方式来理解系统和单个 Service 间的关系并适应基于部件技术的开发方法．

本文提出用基于体系结构的模拟方法来解决 WSS 的可靠性评估问题．主要的难点在于 WSS 具有随机的多用户并行度，这点很少有文献提到．LittleWood[3]曾试图用分析方法考虑 2 用户的情况，对固定的 2 用户的非终止（non-terminating）系统的状态建立可逆马尔可夫过程进行求解．这种情况在实际中非常少见．本文的目标则是要解决随机到来的用户随机被服务后离开的系统的可靠性性质．用模拟方法处理可靠性的相关工作主要有[5~9,12]．文献[5,6]中综述了模拟方法用于可靠性研究的一般思路与技术．文献[7,8,9]提供了把模拟方法应用于实际系统的案例研究．文献[12]提出了一个基于离散事件模拟技术的可靠性工具 SoftRel．但它们没有考虑多用户或动态并行度的情况．本文的主要贡献就是不仅能处理具有动态并行度的系统，也能处理单用户系统的可靠性问题、及模拟文献[4]中提到的基于状态和基于路径的分析模型．

Monte Carlo 方法亦称为随机模拟或统计测试方法．它的基本思想是：建立一个概率模型或随机过程，使它的参数等于

收稿日期：2002-06-10；修回日期：2002-11-05
基金项目：国家自然科学基金（No.60173031）

问题的解;然后通过抽样试验来计算所求解的近似值[10].用 Monte Carlo 方法模拟可靠性过程时,最重要的就是产生各种概率分布的随机变量的子样,如离散型分布、Poisson 过程和基于条件事件速率的过程的抽样等等.由于有各种文献[10]对此作了系统的介绍,限于篇幅,下文不再对 Monte Carlo 抽样技术展开讨论.

下文第 2 节讨论 WSS 的可靠性模型.第 3 节给出模拟系统 Simurel 的实现框架,分析实验结果和运行性能.最后是结论.

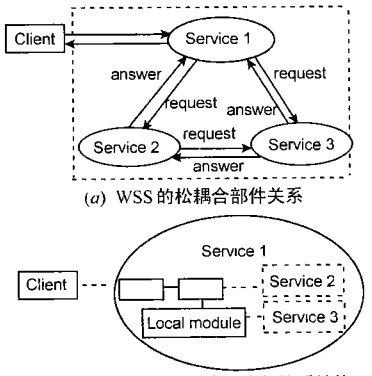
## 2 WSS 的可靠性模型

Web Services 是一种自包含、自描述、模块化的 Web 应用程序.WSS 指通过动态集成 Web Services 的分布式系统.这种系统的显著特点是:高并行度大用户群;自治性的部件(本文中部件与 Web Service 同义使用)松耦合(参图 1).从外部行为的角度来看,一个 Web Service 就如同一个 GI/G/K/K[1] 随机服务系统(Random Server System),K 为允许的最大用户并行度.因此,我们从服务质量的角度定义 WSS 的可靠性,$Rws$,即一个随机到来的请求被正确处理的概率,并且有

$$R_{ws} = P_{acc} \cdot R_{ans}.$$

其中 $P_{acc}$ 是请求被 Web Service 受理的概率,而 $R_{ans}$ 是通常意义上的可靠性度量,即程序产生按规范说明的结果的概率.

从 Web Service 的内部结构考察,它由内部模块和代表外部 Web Service 服务关系的外部模块构成.因此,宜于采用基于体系结构的方法研究 $R_{ans}$.一方面可以计算整体可靠性;另一方面方便进行模块的可靠性规划与优化.模块间的关系可以用执行概图(execution profile)来描述,在有些文献中称为程序流图(Program Flow Graph).执行概图定义了模块及概率形式表示的模块间的控制转移关系.对 WSS 来说,到达的请求可能具有截然不同的执行行为,Simurel 允许同时说明几种执行概图来迎合这种情况.




(a) WSS 的松耦合部件关系

(b) Web Service 1 的集成的内部体系结构

图 1　WSS 的体系结构示例

一般认为失效由模块的失效及模块间控制转移时发生的接口失效组成(失效在本文中的含义可以理解为对给定的输入产生不符合规格说明的输出或输出被无限延时).对 WSS 这种多用户系统,失效发生的后果会有多种:模块性(modular)失效,引起一个线程失效;局部性(local)失效,引起当前共享它的所有线程失效;瞬时性(transient)失效,瞬时或一段时间后失效影响消除;或者是系统性(system)失效,引起整个服务系统崩溃等.失效发生的行为特点也可能不同,如符合一个 Poisson 流、按固定速率发生等.为刻划这些形形色色的失效特点,Simurel 对每一个模块定义了一个失效概图(failure profile),从而允许模块之间的失效多样性,可以适应实际的多种需求.

## 3 Simurel

上一节介绍了解决 WSS 可靠性评估问题的思路,本节讨论原型系统 Simurel 的实现方面的主要问题.

### 3.1 实现框架

如果视一个请求为一个样本,$v$ 为服务成功的请求个数,$N$ 为受理的请求总数,$M$ 为到达的请求总数.按照加强大数定理,当样本量 $N$ 和 $M$ 充分大时:

$$\bar{R}_{ans} = \frac{v}{N} \approx R_{ans}, \bar{P}_{acc} = \frac{N}{M} \approx P_{acc}, \bar{R}_{ws} = \frac{v}{M} \approx R_{ws} \quad (1)$$

式(1)成立的概率等于 1.因此,Simurel 的中心思想就是在 Monte Carlo 随机抽样技术的基础上,建立请求产生器和随机服务器,运行模拟系统足够多的次数或足够长的时间,从而得到统计意义下的可靠性指标.

Simurel 的框架结构如图 2 所示.

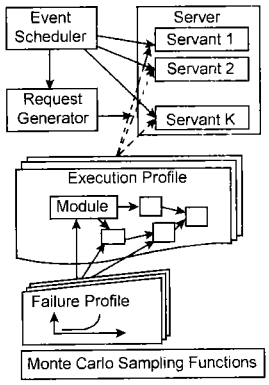



图 2　Simurel 的体系结构

图 2 中,事件调度器 EventScheduler 根据 3.2 介绍的时间机制调度事件,推进模拟进程.服务器 Server 收到请求产生器 RequestGenerator 传递过来的请求时,如果当前并行服务线程 Servant 少于 $K$ 个,则产生一个新的 Servant 处理这个请求,否则请求丢失.每个 Servant 根据请求的类型得到一个相应的执行概图,其中的各个模块均有一个失效概图说明它的执行时间分布和失效的随机性质.请求的产生、模块执行时间、模块间的控制转移及模块失效的产生均通过调用适当的 Monte Carlo 抽样函数完成.

结束模拟的条件可以是到达的或受理的请求达到某个数量,或是模拟时间超过一定长度.

Simurel 是用 Java 语言实现的,它的结构比较简单.但从 3.4 和 3.5 小节的讨论中可以知道,Simurel 是有效而通用的,只要能适当地编写随机分布的抽样,就能模拟多种系统.

### 3.2 时间机制(Timing Mechanism)

Simurel 是连续时间的模拟系统.相对离散事件模拟来说,好处是可以减少离散误差及加快模拟进程.不过,在对某些连续概率分布随机抽样时,为减化抽样过程可能会采取局部离散化方法,如对基于危险率的失效模型就用了半离散化的方法来选取单个失效发生的时间.

模拟系统以事件驱动时钟.事件调度器首先向各个事件源,包括请求产生器和各个服务线程,询问它们的最早发生的事件是在多久之后,如 $t_1, t_2, \cdots$, 从中选出最早的时间 $t' = t_i$,然后通知各事件源将时钟向前拨 $t'$ 个单位,接着触发 $t'$ 发生的事件.由于多用户之间的系统资源共享,服务线程的内部时钟可能比事件调度器维持的时钟慢一些,即需要考虑并行

度为 $k$ 时单个服务线程的减速因子.

一次请求处理的时间由执行概图和模块的执行时间决定.当前模块执行结束后按执行概图中的转移概率选择下一个模块.模块的执行时间则根据实际系统的特点选取最符合要求的分布.这一点也体现了 Simurel 是一种基于体系结构的模拟方法的特点.

这种时间机制提供了在很短的时间内模拟很长时间内的系统变迁的可能性.例如下文实验中的模拟时间可达到现场时间的万分之一以下.

### 3.3 实验结果

为验证 Simurel 模型的有效性,选取具有相应分析模型解的三个可靠性模拟实验.实验平台是:IBM PC 机(主频 700MHz 主存 192MB),Jbulder6 IDE.

**表 1 终止型示例系统的有关参数值**

| 模块 | 转移概率 | 可靠性 $R$ | 执行时间 |
|---|---|---|---|
| 1 | $w_{12}=0.60$ $w_{13}=0.20$ $w_{14}=0.20$ | 0.999 | 100 |
| 2 | $w_{23}=0.70$ $w_{25}=0.30$ | 0.980 | 200 |
| 3 | $w_{35}=1.00$ | 0.990 | 250 |
| 4 | $w_{45}=0.40$ $w_{46}=0.60$ | 0.970 | 100 |
| 5 | $w_{57}=0.40$ $w_{58}=0.60$ | 0.950 | 80 |
| 6 | $w_{63}=0.30$ $w_{67}=0.30$ $w_{68}=0.10$ $w_{69}=0.30$ | 0.995 | 120 |
| 7 | $w_{72}=0.50$ $w_{79}=0.50$ | 0.985 | 180 |
| 8 | $w_{84}=0.25$ $w_{810}=0.75$ | 0.950 | 310 |
| 9 | $w_{98}=0.10$ $w_{910}=0.90$ | 0.975 | 90 |
| 10 | | 0.985 | 150 |

#### 3.3.1 实验一:模拟 Cheung 模型

Cheung 模型是一个被净室工程(Cleanroom Engineering)用于可靠性管理的分析模型.本模拟实验采用与文献[13]相同的案例,即表 1 中说明的执行概图和可靠性指标,但每个模块的平均执行时间 $t_i$ 为[10,60]上的均匀分布的随机子样,模块每次执行的时间则取为均值为 $t_i$ 的负指数分布的随机子样.

**表 2 Cheung 模型的模拟实验结果,文[13]中的分析解 $R=0.8299$**

| 样本量(受理的请求数) | 4 次平均模拟花费时间 | 4 次平均现场运行所需时间 | 模拟解 $\bar{R}$(4 次平均) | 相对误差 $\lvert\bar{R}-R\rvert/R$(%) |
|---|---|---|---|---|
| 3000 | 1675 | 909140 | 0.857 | 3.27 |
| 9000 | 3223 | 3672809 | 0.842 | 1.46 |
| 30000 | 7720 | 13233800 | 0.841 | 1.34 |
| 90000 | 23378 | 31259495 | 0.838 | 0.98 |

#### 3.3.2 实验二:模拟 Littlewood 模型

Littlewood 模型是一个用于非终止型系统的分析模型[4].本实验借用第一个实验中的执行概图,仅把 $w_{101}$ 的转移概率从 0 改为 1,从而变成一个非终止系统.各个模块的失效率均设为 0.025occurrence/ms,模块接口间无失效.则系统的失效率 $\lambda_s=0.025$occurrence/ms.

#### 3.3.3 实验三:多用户环境下的模拟实验

采用与前两个实验相同的执行概图,模块具有与第一个实验相同的可靠性指标.为便于比较平均服务时间,选取各个模块的执行时间为常数值(参表 1).并假设在多用户并行度为 $k$ 时单个服务线程的执行速度减慢的因子为 $f(k)$,$f(k)$ 根据四个预定点(1,1),(20,0.92),(40,0.85)和(70,0.7)通过分段三次样条插值得到.用户请求为平均到达间隔 $1/\lambda=24$ms 的 Poission 流.设最大并发数 $K=20$,样本大小是 3000 个受理的请求数.

**表 3 Littlewood 模型的模拟实验结果**

| | 模块失效总数$\langle A\rangle$ | 模拟花费时间(ms) | 现场运行所需时间(ms)$\langle B\rangle$ | 模拟解 $\lambda_s\langle A\rangle/\langle B\rangle$ | 相对误差 $\lvert\lambda_s-\lambda_s\rvert/\lambda_s$(%) |
|---|---|---|---|---|---|
| | 15889 | 1040 | 600002 | 0.02648 | 5.92 |
| | 15407 | 990 | 600043 | 0.02568 | 2.72 |
| | 15960 | 1370 | 600003 | 0.02660 | 6.40 |
| | 15437 | 990 | 600005 | 0.02573 | 2.92 |
| 平均 | 15673 | 1098 | 600013 | 0.02612 | 4.48 |
| | 160959 | 10650 | 6000014 | 0.02683 | 7.32 |
| | 154782 | 7530 | 6000051 | 0.02580 | 3.20 |
| | 154773 | 8620 | 6000013 | 0.02580 | 3.20 |
| | 154394 | 9280 | 6000009 | 0.02573 | 2.92 |
| 平均 | 156227 | 9106 | 6000019 | 0.02604 | 4.16 |

注:此实验的结束条件是现场运行时间(可以理解为样本量),分别是 10 和 100 分钟.由于最后一个事件发生的随机性,故表中的现场时间不是完全一致.标有"平均"字样的两行是对两种样本量条件下的 4 次实验的平均.

显然地,$C$ 的 $\bar{R}_{ans}=0.8299$ 的精确值是 $R_{ans}=0.8200$,表中的 $\bar{R}_{ans}$ 的实验值是合理的.这个结论同样适用于 $B$ 的 $\bar{R}_{ans}$ 的实验结果.受理同样多的请求(都是 3000 个),$C$ 的现场所需运行时间至少是 $A$ 和 $B$ 的 $\xi$ 倍.$A$ 的平均服务时间应大致为 $C$ 的 $1/f(10.20)=1.0283$ 倍,而 $B$ 的为 $C$ 的 $1/f(11.44)=1.0349$ 倍,上表数据基本符合这个关系.在相同分布的请求流条件下,$A$ 的 $\bar{P}_{acc}$ 一般应比 $B$ 的高,$C$ 的 $\bar{P}_{acc}$ 当然是最差的.由于共享模块的失效牵连影响,$A$ 的 $\bar{R}_{ans}$ 恶化,从而累及 $\bar{R}_{uus}$ 变差.因此在高并行度的环境下,尽量把失效影响局部化或改善高度共享的模块的质量对整个系统的可靠性具有重要意义.由于多用户情况难以得到分析解,通过这三组实验数据交叉验证,可以认为 Simurel 产生的结果是合理的.

另外,从实验结果还可以统计出其它一些有用的数据,例如:

(1)模块被执行的频繁程度,对非终止系统则是指模块的稳态概率,例如从实验二可得出模块稳态概率为{0.1290,0.1180,0.1185,0.0523,0.1743,0.0356,0.0796,0.1140,0.0507,0.1280}.这个分布与分析解{0.1297,0.1177,0.1181,0.0543,0.1751,0.0326,0.0798,0.1133,0.0497,0.1297}比较接近.

(2)在总失效数中各模块所占的比率.

(3)程序一次执行的平均时间等.

**表 4　多用户模型的模拟实验结果**

*A*:模块性失效(Modular Failure)

| 平均服务时间 $1/\mu$(ms) | 1305.34 | 平均并行度 $\bar{\xi}$ | 10.20 | 服务强度 $\rho=\lambda/\mu$ | 54.39 |
|---|---|---|---|---|---|

| 到达的请求数 | $\bar{P}_{acc}$ | $\bar{P}_{ans}$ | $\bar{P}_{us}$ | 模拟花费时间(ms) | 现场运行所需时间(ms) |
|---|---|---|---|---|---|
| 3163 | 0.948 | 0.6911 | 0.6529 | 16200 | 298582 |
| 3156 | 0.951 | 0.6456 | 0.6131 | 16200 | 294472 |
| 3224 | 0.931 | 0.6439 | 0.5983 | 13340 | 300772 |
| 3121 | 0.962 | 0.6129 | 0.5880 | 16370 | 283709 |
| 平均 | 0.948 | 0.6484 | 0.6131 | 15528 | 294384 |

*B*:局部失效(Local Failure)

| 平均服务时间 $1/\mu$(ms) | 1327.58 | 平均并行度 $\bar{\xi}$ | 11.44 | 服务强度 $\rho=\lambda/\mu$ | 55.32 |
|---|---|---|---|---|---|

| 到达的请求数 | $\bar{P}_{acc}$ | $\bar{P}_{ans}$ | $\bar{P}_{us}$ | 模拟花费时间(ms) | 现场运行所需时间(ms) |
|---|---|---|---|---|---|
| 3214 | 0.934 | 0.8264 | 0.7704 | 1260 | 319746 |
| 3151 | 0.952 | 0.8508 | 0.8100 | 1210 | 278677 |
| 3179 | 0.944 | 0.8393 | 0.7870 | 1320 | 331772 |
| 3143 | 0.955 | 0.8242 | 0.7846 | 1270 | 307167 |
| 平均 | 0.946 | 0.8351 | 0.7880 | 1265 | 309341 |

*C*:单并行度

| 平均服务时间 $1/\mu$(ms) | 1275.3 | 平均并行度 $\bar{\xi}$ | 1 | 服务强度 $\rho=\lambda/\mu$ | 53.14 |
|---|---|---|---|---|---|

| 到达的请求数 | $\bar{P}_{acc}$ | $\bar{P}_{ans}$ | $\bar{P}_{us}$ | 模拟花费时间(ms) | 现场运行所需时间(ms) |
|---|---|---|---|---|---|
| 5929 | 0.506 | 0.8416 | 0.4257 | 1050 | 5505891 |
| 5765 | 0.521 | 0.8113 | 0.4222 | 990 | 5613750 |
| 5606 | 0.535 | 0.8170 | 0.4372 | 990 | 5263478 |
| 5720 | 0.525 | 0.8577 | 0.4498 | 980 | 5172622 |
| 平均 | 0.522 | 0.8319 | 0.4337 | 1003 | 5388935 |

注:模块性失效指一个模块发生失效,会引起所有当前共享此模块的服务线程失败.局部失效则仅使触发此失效的服务线程失效.单并行度的情况则退化为实验一的情况.

### 3.4　精度与运行性能分析

从表 2~4 来看,模拟结果的误差在 5% 左右,这样的精度用于通常的设计决策与测试效果评价是可以接受的.受随机误差的干扰,Monte Carlo 方法一般不宜用于求解高精度的问题,这是由这种方法的本质特点决定的.更复杂的抽样方法和其它技术可以在一定程度上补偿精度,这里不再展开这个问题.

由中心极限定理可得统计误差:

$$\varepsilon = |\bar{R} - R| < \frac{\lambda_\alpha \sigma}{\sqrt{N}} \tag{2}$$

近似以概率 $1-\alpha$ 成立,$\alpha$ 为显著水平,$\lambda_\alpha$ 为相对应的正态差.可见 Monte Carlo 模拟的精度阶数为 $O(N^{1/2})$.在实际应用中,如果不能得到理论标准差 $\sigma$,可以用它的估计值:$\sigma(\bar{R}) = \sqrt{\bar{R}(1-\bar{R})/(N-1)}$ 代入式(2)来得到达到要求的精度时所需要的 $N$ 的估计值.

表 5 是实验一的实验误差与按式(2)算出的理论误差的比较.其中的理论标准差 $\sigma = \sqrt{R(1-R)} = 0.375720628$,取 $\lambda_\alpha = 4$,可见实验值与理论值基本成近似关系.然而做模拟实验时,有很多的因素会导致精度的较大随机波动,如表 3 中大样本量得出的结果并不一定总比小样本时的精度高,其中比较明显的影响因素如模拟实验涉及多次或多种随机抽样等.

**表 5　实验误差与理论误差的比较**

| $j$ | $N$ | 实验得出的 $\bar{\varepsilon}$ | $\bar{\varepsilon}_j/\bar{\varepsilon}_{j-1}$ | 按式(2)算出的 $\varepsilon$ | $\varepsilon_j/\varepsilon_{j-1}$ |
|---|---|---|---|---|---|
| 1 | 3000 | 0.0271 | 1 | 0.0274 | 1 |
| 2 | 9000 | 0.0121 | 2.239669 | 0.0158 | 1.734177 |
| 3 | 30000 | 0.0111 | 1.09009 | 0.0087 | 1.816092 |
| 4 | 90000 | 0.0081 | 1.37037 | 0.0050 | 1.74 |

至于 Simurel 的运行性能,与被模拟系统的性质有关.从前面三个实验的结果来看,模拟时间在现场运行所需时间的十分之一到万分之一以下的大范围内变化.主要原因是抽样函数的计算复杂度、事件发生的密集程度等的不同.Simurel 运行时内存的消耗非常少,作者曾经做过一个并行度 = 2000,服务强度 = 650 的实验,包括虚拟机在内的内存需求最高才达到 7MB.

### 3.5　Simurel 应用

通过适当地调整实验参数,Simurel 可以适用于多种情况下的可靠性估算.对非终止型系统如控制系统,只要令执行概图为非可约的(irreducible),如实验二.对终止型系统,可以通过产生一个请求流和适当的并行度 $K$ 值来得到单用户或多用户的版本.

Simurel 的应用范围主要包括:

(1)用模拟方法求解分析模型.按照 Katerina 的分类法,基于体系结构的可靠性评估模型主要分三类:基于状态的,基于路径的,及加成的(additive)模型[4].基于状态和路径的两类模型都可以在 Simurel 中得到模拟近似解,如上文中的 Cheung 模型和 Littlewood 模型;

(2)解决难以用分析模型解决的单用户系统的可靠性问题.Simurel 允许一个系统模型中多种性质的参数存在.例如,在一个系统模型中,可以用正确执行的概率说明一个模块的失效性质,同时用失效率刻划另一个模块的失效行为.这些情况用分析模型方法求解几乎是不可能的;

(3)评估具有固定或动态并行度的多用户系统的可靠性.Simurel 通过说明系统执行概图和每个模块的失效概图,可以模拟非常复杂而微妙的模块共享和失效牵连效果.这是用分析手段难以解决的,也是可靠性模拟的相关工作[7~9,12]所不能处理的.除了可靠性指标,Simurel 还提供了瞬时和统计数据,有助于分析用户流性质、系统结构等对被模拟系统的负载和响应性能的影响.

## 4　结论

本文的最初动机是研究 WSS 的可靠性,但导出的方法及其实现系统 Simurel 超出了原先应用范围.Simurel 通过定义运行概图和失效概图取得了很好的可应用性,可以用于单并行度或多并行度的基于体系结构的可靠性评估,包括模拟基于

状态的、基于路径的分析模型及估算难以导出分析模型的单用户或多用户环境下的可靠性.

Simurel 具有良好的运行效率,其模拟结果的精度对于一般的设计决策或实现评价的用途是可接受的.但高精度场合则需更复杂高级的技术手段.另外,Simurel 除了可以得到可靠性指标,还可以得到另外一些体现系统结构方面的性质的度量数据,如高危模块的识别、平均执行时间的统计值等.

目前的 Simurel 尚未考虑多用户系统对其支持环境的可靠性压力,即由于负载加剧和线程或进程数的增大,导致环境引发的系统失效增多.这里的环境主要是指支持被模拟系统运行的平台,如虚拟机、编程语言的解释器和操作系统等.这将是下一步工作的内容之一,另外一个要做的工作是为 Simurel 编制友好的可视化用户界面.

**参考文献:**


[1] Lyu M R. Handbook of Software Reliability Engineering[M]. McGraw-Hill, 1996.

[2] Mayrhauser A, Malaiya A, et al. On the need for simulation for better characterization of software reliability[A]. Proc., 4th Int'l. Symp. Software Reliability Engineering[C]. 1993. 264 - 273.

[3] Littlewood B, Meyer J F, Wright D R. Dependability of modular software in a multiuser operational environment[A]. Proc. Int' l Symp. Software Reliability[C]. 1995. 170 - 179.

[4] Popstojanova K G, Trivedi K S. Architecture-based approach to reliability assessment of software systems[J]. Performance Evaluation, 2001, 45: 179 - 204.

[5] Gokhale S S, Lyu M R, et al. Reliability simulation of component-based software systems[A]. Proc., 9th Int'l. Symp. Software Reliability Engineering[C]. 1998. 192 - 201.

[6] Tausworthe R C, Lyu M R. A generalized technique for simulating software reliability[J]. IEEE Software, 1996, 13(2): 77 - 88.

[7] Ruggieri M, Cerone E. A simulation tool for the reliability analysis of communication satellite payloads[A]. Simulation'98. Int'l Conference on (Conf. Publ. No. 457)[C]. 1998. 114 - 120.

[8] Salehfar H, Trihadi S. Animated Monte Carlo simulation for teaching power generating system reliability analysis[J]. Education, IEEE Transactions on, 1998, 41(2): 130 - 140.

[9] Goel L, Liang X, O Y. Monte Carlo simulation-based customer service reliability assessment[J]. Electric Power Systems Research, 1999, 49(3): 185 - 194.

[10] 徐钏济.蒙特卡罗方法[M].上海:上海科学技术出版社,1985.

[11] Cerami E. Web Services Essentials[M]. O'Reilly, ISBN: 0 - 596 - 00224 - 6. 2002.

[12] Tausworthe R C, Lyu M R. A generalized software reliability process simulation technique and tool[A]. Proc, 5th Int'l. Symp. Software Reliability Engineering[C]. 1994. 264 - 273.

[13] Cheung RC. A user-oriented software reliability model[J]. IEEE Transactions on Software Engineering, 1980, 6(2): 118 - 125.


**作者简介:**

**支小莉** 女,1974 年生,浙江人,2000 年于浙江大学获工学硕士学位,现为上海交通大学博士研究生,主要研究兴趣:软件可靠性的软件体系结构.

**陆鑫达** 男,1938 年生,上海交通大学计算机系教授、博士生导师,主要研究方向:高性能计算技术和异构网络计算.