# 基于非线性多参数模型的软件老化检测

苏 莉[1,2] 齐 勇[1] 金玲玲[2] 张广路[2]

(西安交通大学电子与信息工程学院 西安 710049)[1] (海南师范大学数学与统计学院 海口 571158)[2]

**摘 要** 提出了一种软件系统的非线性有源自回归(Nonlinear AutoRegressive models with eXogenous Inputs, NARX)网络模型的老化检测方法。解决了目前软件老化方法未考虑多变量间关联性及历史数据的延迟影响的问题。该方法首先通过对实验采集的HelixServer-VOD服务器性能数据进行主成分分析,确定网络的输入维数,根据AIC准则确定最佳模型阶数,最终选取合理的网络模型结构;使用已知的未老化状态样本对NARX网络进行训练,建立系统的辨识模型;然后运用序贯概率比检验(Sequential Probability Ratio Test,SPRT)对NARX辨识模型的残差进行假设检验,判断系统的老化状态。实验分析表明,基于NARX网络模型的故障检测方法能够有效地应用于软件老化的检测。

**关键词** 软件老化,非线性有源自回归网络模型,HelixServer,序贯概率比检验

**中图法分类号** TP301 **文献标识码** A

## Software Aging Detection Based on Nonlinear Multiparameter Model

SU Li[1,2] QI Yong[1] JIN Ling-ling[2] ZHANG Guang-lu[2]

(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)[1]

(School of Mathematics and Statistics, Hainan Normal University, Haikou 571158, China)[2]

**Abstract** This paper presented a method of nonlinear autoregressive models with exogenous inputs to detect the aging phenomenon of the software system. It figures out the problem existing in current software aging methods that there is no cosideration of the correlation between multivariate and the impact of delay from historical data. We first collected the performance data of the HelixServer-VOD server, did principal component analysis of the data, determined the input dimension, determined the best model order according to AIC criteria, selected the reasonable network model structure eventually. We used the known unaging state samples to train the NARX network in order to establish the identification model of the system, then hypothesis tested the residual of the NARX identification model through the method of sequential probability ratio test, finally judge the aging condition of the system. The experimental result shows that NARX model-based fault detection method can be effectively applied in checking software aging.

**Keywords** Software aging, Nonlinear autoregressive models with exogenous inputs, HelixServer, Sequential probability ratio test

## 1 引言

软件老化是软件系统在连续长期运行一段时间后出现性能下降、宕机或崩溃的现象。目前已知的软件老化的原因有操作系统资源枯竭、数据损坏、软件错误累积等。软件老化对软件系统的可靠性是一个严峻的考验。随着基于互联网的各种企业级应用的大量出现,高可靠性和运行时服务质量(QoS)保证问题受到人们的普遍关注。

为了解决以上问题,Huang等人[7]于1995年首次提出软件再生方法,即在系统故障发生之前,通过周期性地暂停软件的运行,清除错误累积的环境,在干净的初始状态或中间状态下重新启动。再生时间的选择有两种,一种是简单地按照某个固定的时间间隔重启系统,另一种是基于预测的方法。而后者是更加有效的方法。该方法需要一些性能的度量标准,通过对系统的监测来发现老化的开始[9]。目前基于后者的研究主要分为基于模型的和基于测量的两种方法。基于模型的方法需要对软件系统的状态关系建立模型,主要有马尔科夫模型[7]、半马尔科夫模型[11]、马尔可夫决策过程模型[12]、随机Petri网[13]等数学模型。以此计算软件系统最优的再生周期或再生策略。此方法的缺陷是这些随机数学模型的一些基本假设往往过于理想,并不易在工程领域内被验证[14]。基于测量的方法则更侧重于反映真实实验数据的规律,通过监测软件系统的运行状况并采集关键性能参数,然后利用回归技术或者机器学习算法分析软件的老化程度和判断再生动作执行点。目前主要的分析技术包括局部线性回归、决策树、神经网络、支持向量机等。

到稿日期:2012-03-29 返修日期:2012-08-20 本文受国家自然科学基金(60933003),海南省自然科学基金(610221)资助。

苏 莉(1982—),女,博士,讲师,主要研究领域为软件老化与再生;齐 勇(1957—),男,博士,教授,主要研究领域为分布式与并行计算;金玲玲(1976—),女,硕士,讲师,主要研究领域为数据库系统技术、分布式计算,E-mail:50799003@qq.com(通信作者)。

针对目前关于软件老化分析主要为单参数模型，没有考虑多参数间的关联和影响性，以及历史数据的延迟影响的问题，本文提出了一种基于 NARX 网络模型的方法，它在首次进行软件老化分析的同时，考虑到关联多变量、有反馈、有延迟输入的非线性回归方法。利用 NARX 网络模型建立系统正常状态下的辨识模型，通常情况下 NARX 回归神经网络性能优于全回归神经网络，并且可以和全回归神经网络互相转换[2]，当系统处于老化状态时，辨识模型的输出残差会偏离正常水平[1]。使用序贯概率比检验(Sequential Probability Ratio Test，SPRT)对辨识模型的残差进行假设检验，从而实现系统的老化检测。通过与单参数 NARX 模型、高维 NARX 模型及向量自回归模型进行比较，证实了本文给出的基于 NARX 网络模型的方法能更准确地模拟软件运行状态趋势，检测软件老化现象。

## 2 非线性有源自回归网络模型

NARX 模型是一类重要的离散时间非线性系统，即

$$y(t)=f(u(t-n_u),\cdots,u(t-1),u(t),y(t-n_y),\cdots,y(t-1)) \quad (1)$$

式中，$u(t)$和 $y(t)$分别表示 $t$ 时刻的系统输出和输入；$n_u$ 和 $n_y$ 分别表示输入和输出的阶数；$f(\cdot)$是非线性函数，当用多层感知器(Multilayer Perceptron，MLP)产生的映射关系来逼近函数 $f(\cdot)$时，可将式(1)改写为

$$y(t)=\Psi(u(t-n_u),\cdots,u(t-1),u(t),y(t-n_y),\cdots,y(t-1)) \quad (2)$$

式中，$\Psi$ 表示由 MLT 刻画的非线性映射。此模型是一种将有源自回归(auto-regressive with extra inputs，ARX)模型与神经网络相结合的系统建模方法，被证明与图灵机具有相同的计算辨识能力[2]。图 1 所示为一个三层神经网络，输入层由信号源结点组成，第二层为隐含层，第三层为输出层，从输入层到隐含层的变换函数是非线性函数，从隐含层到输出层的变换是线性变换[2]。

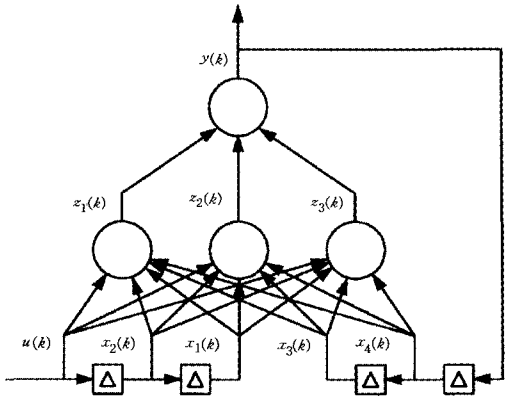



图 1 $n_u=n_y=2$，$H=3$ 的 NARX

神经网络的输入状态更新如下

$$x_i(t+1)=\begin{cases}u(t), & i=n_u\\ y(t), & i=n_u+n_y\\ x_{i+1}(t), & 1\leqslant i\leqslant n_u \text{ and } n_u\leqslant i\leqslant n_u+n_y\end{cases}$$

$t$ 时刻输入对应的状态向量如下

$$x(t)=[u(t-n_u)\ \cdots\ u(t-1)\ y(t-n_y)\ \cdots\ y(t-1)]$$

$H$ 为隐层节点个数，则隐含层节点的激活函数定义如下

$$Z_i(t)=\sigma\left(\sum_{j=1}^{N}a_{i,j}x_j(t)+b_iu(t)+c_i\right),\ i=1,\cdots,H$$

$\sigma$ 为非线性激活函数。

输出层仅包含一个线性节点，即

$$y(t)=\sum_{j=1}^{H}\omega_{i,j}z_j(t)+\theta_i$$

式中，$\omega_{i,j}$表示权重，$\theta$ 表示偏差。

## 3 序贯概率比检验

与阈值监测方法相比，序贯概率比检验是一种更灵敏的状态监测方法。在众多的统计检验方法中，序贯概率比检验做出信号正常与否的决策时所需的平均样本数最小。这就意味着，在同样的精度下，序贯概率比检验能够给出发展性故障信号的最早指示。序贯概率比的显著性水平、功效函数都可以通过近似计算得到，由于这种检验具有一般性和简单性，自其提出以来，已在工业控制领域、信号处理和医药检测领域得到了广泛的应用。

序贯抽样方案是指在抽样时不事先规定总的抽样个数(观测或实验次数)，而是先抽少量样本，根据其结果再决定停止抽样或继续抽样、抽多少，这样下去，直至决定停止抽样为止。第二次世界大战时，为军需验收工作的需要，瓦尔德发展了一种一般性的序贯检验方法，叫序贯概率比检验(简称 SPRT)。其要点如下：设在原假设 $H_0$ 和备择假设 $H_1$ 之下，随机变量 $x$ 的概率密度函数或概率函数随机变量都已知，且分别为 $p_0(x)$及 $p_1(x)$，对 $x$ 逐次观测，第 $i$ 次观测的结果记为 $x_i$，称比值

$$\lambda_n=\lambda_n(x_1,x_2,\cdots,x_n)=\prod_{i=1}^{n}\left(\frac{p_1(x_i)}{p_0(x_i)}\right)$$

为样本 $x_1,x_2,\cdots,x_n$ 的概率比。在固定抽样方案之下，事先给定自然数 $n$，对 $x$ 进行 $n$ 次观测得 $x_1,x_2,\cdots,x_n$，计算 $\lambda_n=\lambda_n(x_1,x_2,\cdots,x_n)$。指定两个数 $A,B,A<B$，若 $\lambda_n\leqslant A$，则接受 $H_0$；若 $\lambda_n\geqslant B$，则接收 $H_1$(拒绝 $H_0$)；若 $A<\lambda_n<B$，则继续抽样一次得 $x_{n+1}$，计算出 $x_{n+1}$再做上述比较，直到做出决定为止。$A,B$ 的定法则取决于指定的两种错误概率 $\alpha$ 和 $\beta(\alpha,\beta$ 都大于 0，但很小)。

## 4 实验环境描述与数据采集

HelixServer 是一款主流的流媒体应用服务器软件，支持 RTSP，RTP，HTTP，MMS 等多种流媒体协议。由于多格式、跨平台和开源性等优点，HelixServer 应用十分广泛，且源代码数量庞大，属于有代表性的复杂系统软件，因此选取 HelixServer 作为老化实验的研究对象。利用 RTSP 协议以及 RDT 协议开发了 HelixClientEmulator 用于模拟大量客户端访问服务器上的流媒体文件。HelixClientEmulator 采用多线程机制，可以真实模拟多个客户端同时访问流媒体，每个客户端包括 3 个线程，一个线程用于处理音频流，一个线程用于处理视频流，还有一个线程用于 session 管理包括：建立连接、中断连接以及 QoS 参数统计等。实验平台包括 3 台客户端，1 台 Helix Server 服务器，及 1 台千兆带宽的交换机，构成小型局域网络。实验平台如图 2 所示。HelixServer 上分布有 100 个不同等级的 rmvb 文件。采用 DOE 方法产生不同的负载，负载由元组(client_count，file_ploy，file_object，file_max_object，sleep_time，file_difference)表示。具体描述见表 1。

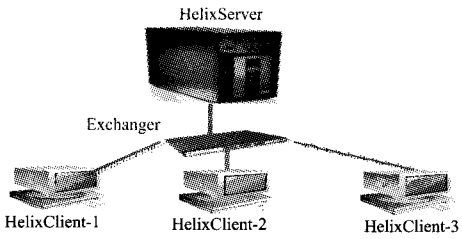



图 2　实验平台拓扑关系

表 1　负载属性及其物理意义

| 属性 | 物理意义 |
|---|---|
| client_count | 模拟的客户端数目 |
| file_ploy | 客户端访问文件模式，HelixClientEmulator 可以模拟 3 种访问模式分别是：0 随机访问；1 顺序访问；2 Possion 分布访问；3 单文件访问 |
| file_object | 0-FileMaxObject 范围内允许访问的文件数 |
| file_max_object | 100 个流媒体文件中最大允许访问的文件数量 |
| sleep_time | 请求播放完后到发起新的请求之间的间隔(Milliseconds) |
| file_difference | 文件大小是否相同，0 表示不同，1 表示相同 |

实验中 client_count 恒定设置为 600(经服务器容量测试最大连接数为 900，因此不会因为过负载而导致伪老化)。HelixclienEmulator 参数集中的数据采集周期是不固定的，其他参数都是每隔 15s 采集一次，实验持续时间为 291480s，约合 81 小时，共采集样本 19432 个。实验中所采集的参数(见表 2)可以很全面地描述 CPU、内存、I/O 操作以及 VOD 服务器网络带宽等计算机系统基本资源的使用状况。

表 2　实验采集的参数

| level | Parameter content |
|---|---|
| Client | Client Num<br>Jitter in last 10 seconds<br>Total jitter<br>Average bandwidth<br>Average response time |
| OS | \Memory\Available Mbytes<br>\Memory\Cache Bytes<br>\Memory\Cache Faults/sec<br>\Memory\Page Faults/sec<br>\Memory\Page Reads/sec<br>\Memory\Pool Paged Bytes<br>\Memory\SystemCacheResidentBytes<br>\PhysicalDisk(_Total)\% Disk Time<br>\PhysicalDisk(_Total)\Avg. DiskQueueLength<br>\Processor(_Total)\% Interrupt Time |
| Helix Server | Players Connected<br>Average Bandwidth Output Per Player(bps)<br>Total Actual Bandwidth Output(bps)<br>Memory Stats(Bytes In Use)<br>LoadState |
| Rmserver process | \Process(rmserver)\% Privileged Time<br>\Process(rmserver)\% User Time<br>\Process(rmserver)\IO Data Bytes/sec<br>\Process(rmserver)\Virtual Bytes<br>\Process(rmserver)\Working Set<br>\Process(rmserver)\Private Bytes |

## 5　软件老化的 NARX 网络模型的建模与检测

### 5.1　数据的预处理及主成分分析

由于实验采集的数据有噪声且不一致，因此首先需要对数据进行清理，即填写缺失的值、光滑噪声数据、识别或删除离群点并解决不一致性。本实验中采用移动平均数法对所有数据进行平滑处理。

实验中采集的参数如表 2 所列，由于采集参数维数较多，多种参数的单位不一致(如 Memory 的 Available Mbytes 是以兆字节为单位，而 Memory 的 Cache Bytes 则以字节为单位，服务器的 Total Actual Bandwidth Output 以 bps 为单位)，其被称为奇异数据。奇异样本数据的存在会引起神经网络训练时间的增加，并可能引起网络无法收敛，所以在训练之前对训练样本数据集进行了归一化处理，以便后面数据的处理，并保证程序运行时收敛加快。

以 2011 年 6 月 22 日 14:51:11 至 15:40:11 的 198 个样本数据为例，取 Memory 的 Pages/sec 及 PhysicalDisk(_Total)的 Avg. Disk Queue Length 作为网络输入，Memory 的 Available MBytes 作为网络输出，采用 5 阶延迟，分别对预处理前后的数据进行训练，将训练结果与实际数据做回归处理，数据未作预处理前所产生的结果回归系数为 0.92722，预处理后所产生的结果的回归系数为 0.99832。从以上两个数据可以看出，在本实验环境下，数据预处理可以明显地提高实验数据的质量，并有利于网络的收敛。

实验过程中收集了 30 个内存参数、15 个 CPU 参数和 27 个应用服务器参数。排除零方差参数及具有相同意义但单位不同的变量后，还剩下 40 个参数。如果 40 个参数全部用来做 NARX 模型的创建，模型规模较大，训练时间较长。因此，采用主成分分析方法选取 $n(n<40)$ 个具有代表性的变量来解释大部分的数据变化情况。实验中采用最大四次方值旋转方法，共提取出 10 个主成分，累计贡献率为 88.686%。通过对旋转成分矩阵的分析，得出其中 16 个变量能够反映原有的绝大部分信息。在给定显著性水平下，选取了 4 个影响最为显著的变量作为模型的输入输出。其中 Memory\Available Bytes、Memory\Cache Bytes 和 Memory\Pages/sec 作为 NARX 模型的输入，分别由 $U_1$，$U_2$，$U_3$ 表示，Processor(_Total)\Privileged Time 作为模型的输出，由 $y$ 表示。因此可以得出 NARX 网络方程为：

$$Y(t)=\Psi(y(t-1)+\cdots+y(t-m)+U_1(t-1)+\cdots+U_1(t-n_1)+U_2(t-1)+\cdots+U_2(t-n_2)+U_3(t-1)+\cdots+U_3(t-n_3))$$

式中，$m$，$n_1$，$n_2$，$n_3$ 分别是输入输出的延时阶数。

### 5.2　隐层阶数和延时阶数的选取

一般认为，增加隐层数可以降低网络误差，提高精度，但也使网络复杂化，从而增加了网络的训练时间和出现“过拟合”的倾向。由于复杂系统软件具有较高的非线性度，单隐层结构无法满足要求，因此在实验中选择隐层数为 2，每层神经元个数为 10。

模型的阶数确定可以根据 AIC(Akaike's Information Criterion)准则进行判定，它是数字信号处理中对多种模型做选择的判别方法。其定义为 $AIC=-2\lg$(模型的最大似然度)$+2$(模型的独立参数个数)$=-2L(\hat{\beta})+2k$。式中，$k$ 为独立参数个数，$\hat{\beta}$ 为参数的最大似然估计值，$L(\cdot)$ 为似然函数。考虑到 NARX 模型是一类重要的离散时间非线性系统模型，我们认为误差 $\varepsilon_i$ 相互独立、方差同质服从正态分布[2]。模型的选择实际上可用 $AIC=\ln(V)+2p$ 来完成，其中，$V$ 可用模型残差的方差代替，$p$ 是模型中独立参数维数。计算 $\arg\min(AIC)$ 可以得出 NARX 模型的最佳阶数。

### 5.3 NARX网络模型的实验分析

序贯概率比检验需要系统处于正常和老化两个状态的数据。正常工作状态下的数据用于NARX建模及老化检测的测试;老化状态下的数据作为老化检测测试样本。因此需要选取合适的数据。本文采集的实验观测样本为系统负载600,1,100,100,0,1000,持续运行11.5个小时的实验结果。首先,HelixServer参数集中的Average Bandwidth Output Per Player参数,每个播放器的平均带宽在经历快速下降后逐渐趋近稳定,但是仍存在下降趋势;HelixServer耗用的内存总量及页面出错率在实验过程中有缓慢上升趋势,OS参数集中的system cache resident byte参数显示系统缓冲的文件大小一直在增加,而处理器利用率和输出网络带宽则有缓慢下降趋势,这反映了软件系统性能在下降,可以判定系统发生了老化,因此可以收集到两个状态的数据。HelixServer的空闲内存(见图3)在实验初期变化明显,而在后期变化不显著,在大约437个观测点后出现了明显的下降趋势。因此选取前298个观测点的数据作为正常状态数据,来训练NARX网络。选取系统的实际输出和模型的预测输出之间的差别为残差,从图4(a)可以看出,绝大部分残差在±0.05之间,图4(b)中显示其回归系数为0.96696,训练时间为12分11秒。为了防止其受到极端数据的影响,在同一系统中重复试验20次,平均回归系数为0.970287,平均训练时间为12分26秒。

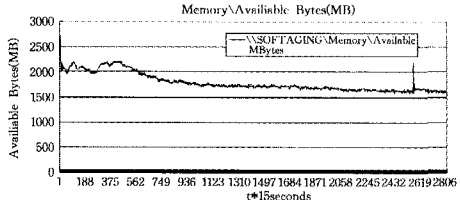
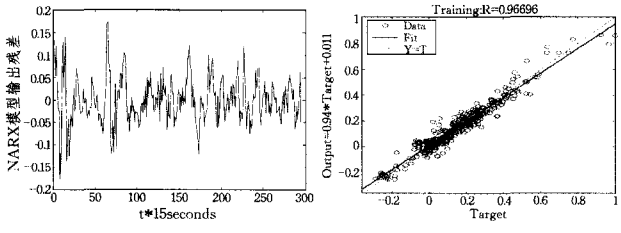



图3 Memory\Available MBytes




(a) NARX模型残差 (b) NARX模型回归图

图4

### 5.4 基于序贯概率比的老化检测

若系统参数变化导致残差值发生零偏,设残差均值为$\mu$,则设定正常状态$H_0:\mu=0$;故障状态$H_1:\mu=\mu_0\neq0$。定义系统的残差序列向量为$r(t)=[r(1),r(2),\cdots,r(t)]$,通过对$r(k)$做Kolmogorov-Smirnov检验,检测出残差是服从正态分布的。随机序列的联合概率密度函数为:

$$P=\frac{1}{(\sigma\sqrt{2\pi})^2}\exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-u_0)^2\right]$$

将$\lambda_n$两边取自然对数,用已知变量$x$的先验方差$\sigma^2$代替样本总体方差,整理后,得:

$$\lambda(n)=\frac{n}{2}\times\frac{(\frac{1}{n}\sum_{i=1}^{n}x_i-x_0)^2}{\sigma^2}$$

假设将$H0$判定为$H1$的概率(称为误报警率)为$\alpha$;将$H1$判定为$H0$的概率(称为漏报警率)为$\beta$,从而可以得到序贯概率比检验的阈值$A=\ln\frac{1-\alpha}{\beta}$,则老化的判断规则为:

$$\begin{cases}\lambda(n)\geq A, & \text{则判定 } H1 \text{ 为真,发生老化}\\ \lambda(n)<A, & \text{继续检测}\end{cases}$$

从图3和图5可以看出,从749观测点后老化现象明显。取$\lambda_0=0$,900到2000观测点数据的残差序列计算其标准差为0.24061,均值为−0.92376。取从1到200观测点数据的残差的均值−0.17378作为$x_0$。取$\alpha=0.01,\beta=0.01$,计算得出检测阈值分别为$A=4.595$。图6为5.1节模型的老化判断曲线,第560个观测点$\lambda(n)>A$,从此处开始判断出现了老化现象。

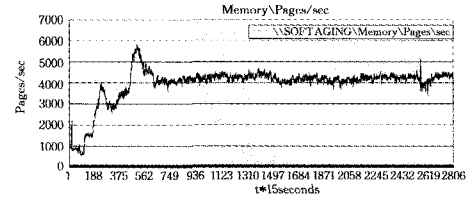



图5 Memory\Available MBytes

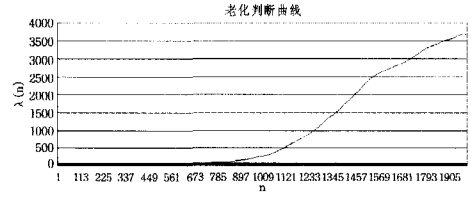
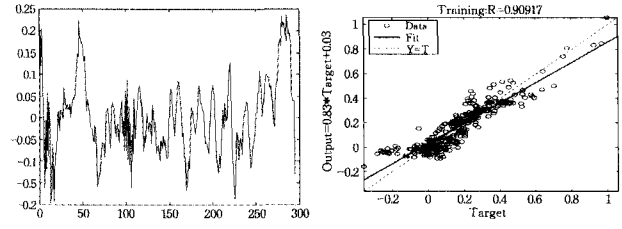



图6 5.1节模型中老化判断曲线

## 6 模型对比

### 6.1 单变量NARX模型

为了对比NARX模型的精度,首先使用NARX的单参数模型进行对比,选取Memory\Available Bytes为输入,Processor\ Privileged Time为输出,隐层阶数和延时阶数不变,训练的观测点个数和数据内容不变,数据仍然进行了预处理,此次训练时间为4分9秒。对比图7(a)和图4的残差可知,多变量NARX模型比单变量NARX模型模拟效果更好;从图7(b)可以看出,单参数NARX模型回归系数为0.90917,同样重复试验20次,平均回归系数为0.92696,平均训练时间为12分15秒,回归系数比多参数模型回归系数下降,训练时间减少了约11秒。



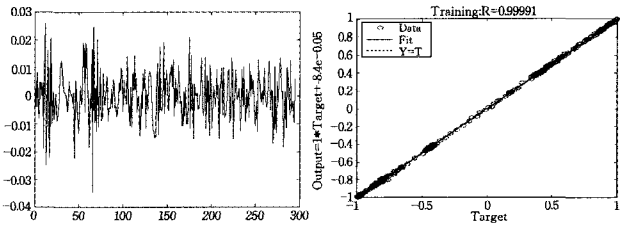

(a) 单变量NARX模型残差 (b) 单变量NARX模型回归图

图7

### 6.2 高维NARX模型

为了观察NARX模型维数与精度的关系,我们进行了较高维的数据试验。通过主成分分析,将能够反映原有的绝大部分信息的16个变量全部用来作为NARX模型的输入输出,使用同时期预处理后的数据进行训练模拟,结果的残差和回归图如图8所示。同样重复试验20次,结果为平均回归系

数是 0.999906，平均试验时间为 14 分 39 秒。从试验结果可以看出，NARX 的维数越高，模拟效果越好，但试验时间也随之增加。




(a) 高维 NARX 模型残差　　(b) 高维 NARX 模型回归图

图 8

### 6.3 向量自回归模型(Vector Autoregression, VAR)

向量自回归(VAR)是基于数据的统计性质建立模型，VAR 模型把系统中每一个内生变量作为系统中所有内生变量的滞后值的函数来构造模型，从而将单变量自回归模型推广到由多元时间序列变量组成的"向量"自回归模型。VAR 模型是处理多个相关经济指标的分析与预测最容易操作的模型之一，并且在一定的条件下，多元 MA 和 ARMA 模型也可转化成 VAR 模型。VAR($p$) 模型的数学表达式为：

$$y_t = A_1 y_{t-1} + \cdots + A_p y_{t-p} + BX_t + \varepsilon_t$$

式中，$y_t$ 是 $k$ 维内生变量向量，$X_t$ 是 $d$ 维外生变量向量，$p$ 是滞后阶数，样本个数为 $T$。$k \times k$ 维矩阵 $A_1, \cdots, A_p$ 和 $k \times d$ 维矩阵 $B$ 是要被估计的系数矩阵。$\varepsilon_t$ 是 $k$ 维扰动向量，它们相互之间可以同期相关，但不与自己的滞后值相关且不与等式右边的变量相关。

Vector Autoregression Estimates
Date: 03/28/12 Time: 16:12
Sample (adjusted): 5 300
Included observations: 296 after adjustments
Standard errors in ( ) & t-statistics in [ ]

| | PROCESSO... |
|---|---|
| PROCESSOR_T(-1) | 0.512982<br>(0.05770)<br>[8.89002] |
| PROCESSOR_T(-2) | 0.129676<br>(0.06533)<br>[1.98505] |
| PROCESSOR_T(-3) | 0.207768<br>(0.06062)<br>[3.42747] |
| MEMORY_AVAIL | 3.77E-09<br>(1.8E-09)<br>[2.13264] |
| MEMORY_CACHE | -3.98E-08<br>(2.7E-08)<br>[-1.49485] |
| MEMORY_PAGES | 0.000185<br>(0.00011)<br>[1.69063] |
| R-squared | 0.538732 |
| Adj. R-squared | 0.530779 |
| Sum sq. resids | 555.6059 |
| S.E. equation | [illegible] |
| F-statistic | 67.74031 |
| Log likelihood | -513.2014 |
| Akaike AIC | 3.508117 |
| Schwarz SC | 3.582922 |
| Mean dependent | 11.90993 |
| S.D. dependent | 2.020671 |

图 9　5.1 节中参数的向量自回归模型分析

EViews 是在大型计算机的 TSP (Time Series Processor)软件包基础上发展起来的新版本，是一组处理时间序列数据的有效工具。根据第 5.1 节中主成分分析的结果，使用 EViews 软件进行向量自回归模型建模、估计和预测，使用最小二乘法估计方法对 3 到 300 观察点的数据进行无约束向量自回归建模，其中 Memory\Available Bytes、Memory\Cache Bytes 和 Memory\Pages/sec 3 个量作为外生变量，Processor(_Total)\Privileged Time 作为内生变量，滞后信息选取为 3 阶。图 9 为本次估计结果。对比图 10 及图 4 的残差图，说明线性模型的拟合效果并没有 NARX 模型模拟效果理想。

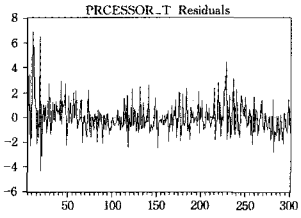



图 10　向量自回归模型分析的残差

**结束语**　本文提出了一种针对软件系统的基于非线性有源自回归网络模型的老化检测方法。NARX 模型考虑关联多变量、有反馈、加入延迟输入，并使用非线性回归方法。进一步可以将此模型视为一个胶囊单元，在复杂的系统中使用，将不同粒度、不同子系统或层次作为一个胶囊建模。鉴于 NARX 模型优于全回归神经网络模型，胶囊思想可以更加契合复杂的系统本身。我们分别与单变量 NARX 模型和向量自回归模型进行了对比，从结果可知，NARX 模型的多变量和非线性使之与软件系统本身契合度较高。数值仿真表明，基于 NARX 神经网络的辨识器具有很强的非线性动态描述能力，解决了目前软件老化方法未考虑多变量间关联性及历史数据的延迟影响的问题。下一步的工作重点是进一步完善模型设置，调整模型的输入输出，分辨性能下降是由于老化还是 workload 的改变所导致的；另一方面，设计针对实时动态数据进行模拟预测。

## 参 考 文 献


[1] Tian Y. Static study of the porous bearings by the simplified finite element analysis[J]. Wear, 1998, 218: 203-209

[2] Siegelmann H T, Horne B G, Giles C L. Computational Capabilities of Recurrent NARX Neural Networks[J]. IEEE Transactions on Cybernetics, 1997, 27(2): 208-215

[3] 蒋浩天，拉塞尔 E L，布拉茨 R D. 工业系统的故障检测与诊断[M]. 段建明，译. 北京：机械工业出版社，2003

[4] 孟海宁，齐勇，侯迪. 基于非马尔可夫随机 Petri 网的软件再生建模与分析[J]. 计算机学报，2007，30(12)：2212-2217

[5] 苏浩秦，宋述杰，邓建华. 基于限制最小二乘估计的飞机舵面故障诊断方法[J]. 机械科学与技术，2005，24(9)：1033-1035

[6] Kourai K, Chiba S. A Fast Rejuvenation Technique for Server Consolidation with Virtual Machines[C]// IEEE/IFIP International Conference on Dependable Systems and Networks. Edinburgh, UK, 2007: 245-255

[7] Huang Y, Kintala C, Kolettis N, et al. Software Rejuvenation: Analysis, Module and Applications [C] // Proc. 25th Symp. FaultTolerant Computing. Pasadena, USA: IEEE Press, 1995: 381-390

[8] 易丹辉. 数据分析与 Eviews 应用[M]. 北京：中国人民大学出版社，2008：31-50

[9] Trivedi K S, Vaidyanathan KGo˘seva-Popstojanova K. Modeling and Analysis of Software Aging and Rejuvenation[C]// Proc. of ANSS-2000. April 2000


(下转第 170 页)

(1)获取初始的软件维护场景

通过与各个风险承担者的沟通，按照各个风险承担者(角色)划分，获得的初始维护场景如表1所列。

表1 初始的软件维护场景

| 序号 | 角色 | 维护场景描述 | 权重 |
|---|---|---|---|
| $s_1$ | 用户 | 增加对多种语言源程序故障诊断的支持 | 0.3 |
| $s_2$ | | 添加实时诊断功能 | 0.10 |
| $S_3$ | 系统管理员 | 增加日志管理 | 0.05 |
| … | | … | |

(2)软件维护场景表示

以维护场景 $s_1$ 为例来说明本文方法的具体应用。第1个维护场景是用户提出的，其目的是使系统支持更多语言的源代码分析和故障诊断(原来只支持C和C++语言)，现在要增加对Java、VB等语言的支持功能。通过变更传播分析($s_1$→用例确定源程序分析器→SourceAnalysis模块)可以看出，要实现维护场景 $s_1$，需要对构件SourceAnalysis进行更改，由图5系统的组件图可以得到它的邻接关系矩阵 $M$，并求出它的可达矩阵 $M^+$。

$$M=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&1&0&0\end{bmatrix}\quad M^+=\begin{bmatrix}1&1&1&0&0\\0&1&1&0&0\\0&1&1&0&0\\0&1&1&1&1\\0&1&1&0&1\end{bmatrix}$$

由可达矩阵 $M^+$ 可以看出，对构件SourceAnalysis($C_2$)和KnowledgeMaintain($C_3$)的更改将影响到其它所有的组件。通过上述分析，我们可以将维护场景 $s_1$ 表示为：

(增加多种语言的支持，Action＝(Change_C，SourceAnalysis component，($C_1$，$C_2$，$C_3$，$C_4$，$C_5$))，5)

用三模板[6]来表示由场景 $s_1$ 构建的维护性需求模型如下：

```
attribute Expandability parent maintainability
{
  component system,SourceAnalysis component
  children Modularity,Flexibility
  contribution oneX
}
  Constraint good_ Expandability
{
  constraints
  Modularity [strong ]
  Flexibility [strong ]
…
  priorities
  Modularity [high]
  Flexibility [high]
}
Operation add SA_LanguageSelect Class
{
  affected Modularity
  implemented Flexibility
  effect
  Modularity [+3]
}
```

**结束语** 本文针对当前软件维护性需求获取困难、缺乏维护性分析设计方法的难题，提出了一种基于软件维护场景的维护性需求分析方法，该方法能够帮助软件需求分析和设计人员在软件开发早期把握软件维护性需求，进而设计赋予软件良好的维护性。实例应用表明，该方法能够为开发人员提供一种较好的维护性需求分析方法，我们将在下一步工作中对方法进行深入实践和不断改进。

## 参考文献


[1] Bosch J. Design & Use of Software Architectures: Addison Wesley[M]. 2000

[2] Bass L, Clements P, Kazman R. Software Architecture in Practice: Addison-Wesley[M]. 1998

[3] Mylopoulos J, Chung L, Nixon B A. Representing and using nonfunctional requirements: a process-oriented approach [J]. IEEE Transactions on Software Engineering, 1992, 18(6): 483-497

[4] Firesmith. OPEN Process Framework (OPF) [OL] http://www.donald-firesmith.com, 2004

[5] Peters J F, Pedrycz W. Software Engineering, an Engineering Approach[M]. Wiley, 2000

[6] 叶飞，朱小冬，王毅刚. 基于XML的软件非功能需求建模研究[J]. 微计算机信息，2008，24(3)：250-252

[7] 王映辉，张世琨，刘瑜，等. 基于可达矩阵的软件体系结构演化波及效应分析[J]. 软件学报，2004，15(8)：1107-1115


(上接第165页)


[10] Grottke M, Nikora A, Trivedi K. An Empirical Investigation of Fault Types in Space Mission System Software[C]// IEEE/IFIP International Conference on Dependable Systems and Networks. Chicago, USA, 2010: 447-456

[11] Dohi T, Goseva-Popstojanova K, Trivedi K S. Statistical nonparametric algorithms to estimate the optimal software rejuvenation schedule[C]// Proceedings of the Pacific Rim International Symposium on Dependable Computing (PRDC2000). Los Angeles, USA, 2000: 77-84

[12] Pfening A, Garg S, Puliafito A, et al. Optimal software rejuvenation for toleration software failures[C]// Proceedings of the 18th International Symposium on Performance Evluation. Lausanne, Switzerland, 1996: 491-506

[13] Garg S, Huang Y, Kintala C, et al. Time and load based software rejuvenation: Policy, evaluation and optimality[C]// Proceedings of the 1st Fault Tolerance Symposium, FTS-95. Madras, India, 1995: 22-25

[14] Andrzejak A, Silva L. Robust and adaptive modeling of software aging[OL]. http://citeseerx.ist.psu.edu/viewdoc/summary, 2011-07

[15] Bozdogan H. Model selection and Akaike's information criterion (AIC): the general theory and its analytical extension [J]. Psychometrika, 1987, 52: 345-570

[16] Controneo D, Natella R, Pietrantuono R, et al. Software aging analysis of the Linux Operating System[C]// 2010 IEEE 21st International Symposium on Software Reliability Engineering. San Jose CA, USA, 2010: 71-80