一种软件可靠性增长模型选择与综合方法

冯光成 顾 庆 陈 翔 陈道蓄
(南京大学计算机科学与技术系计算机软件新技术国家重点实验室 南京 210093)

摘 要 软件可靠性增长模型可以预测软件在将来某个时刻的可靠性,以此作为软件是否发布的依据。而目前常见的各种模型对不同失效数据集的预测能力并不一致。提出了一种软件可靠性增长模型选择和应用的框架,利用可靠性模型评价准则,对特定的失效数据集选择优选模型集,根据优选模型集利用神经网络较好的学习预测能力计算可靠性。利用此方法对实际软件项目中的失效数据进行了分析,并验证了它的有效性。

关键词 软件可靠性,可靠性增长模型,模型选择与综合,神经网络

Software Reliability Growth Model Selection and Composition Method

FENG Guang-cheng GU Qing CHEN Xiang CHEN Dao-xu
(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract Software reliability growth models are used for the prediction of software reliability which can decided whether software could make release. At present, all kinds of common models' predict ability is inconsistent. We proposed a framework for software reliability growth model selection and application. The method provided guidelines on how to select better model set on the given failure data set, and then used these models to predict reliability with neural network. The method was applied in a case study using failure data from some real world software projects and the experiment indicates its efficiency.

Keywords Software reliability, Software Reliability Growth Model, Model selection and compose, Neural network

1 引言

软件可靠性是指在给定的执行环境下,软件持续执行一段时间间隔或一定数量事物单元(如打印的页数或ATM软件的存取操作)无失效运行的概率[1]。软件可靠性是软件系统的固有特性之一,表明了一个软件系统按照用户的要求和设计的目标、执行其功能的正确程度。在软件开发过程中,可以通过测试来度量软件可靠性。软件可靠性工程对可靠性设置定量目标,并通过制定策略来达到这些目标。软件可靠性工程技术目前已在业界采用,包括AT&T, Lucent, IBM, NASA, Microsoft等公司[2]。

软件可靠性增长模型(SRGM, Software Reliability Growth Model)是预测和评估软件可靠性强有力的工具。1972年,Jelinski和Moranda提出了第一个模型JM[3],迄今为止已经提出了不少于100种的模型,但还没有一种通用的模型能应用到所有的软件项目中。对于不同的软件,甚至是软件的不同版本都会有不同的适用模型。潜在的问题是因为影响软件可靠性的原因比较复杂,主要包括[4]测试类型、软件性质、人为因素和调试过程。

有于此,需要一种方法应用不同的可靠性增长模型对软件的失效数据进行建模,然后通过特定的策略选择最佳的可靠性增长模型来计算软件的可靠性指标,从而对未来的测试过程做有效的预测。

本文首先简要介绍了软件可靠性增长模型,然后提出一种模型选择和综合应用的框架,并进行了案例分析,最后给出总结并提出展望。

2 软件可靠性增长模型

2.1 软件可靠性工程

软件可靠性工程(SRE: Software Reliability Engineering)[5]是以软件一系列重要特性为中心而开展的,它定量地按用户对于可靠性的需求,进行研究基于软件系统的操作行为。它被证明为是一种较好的实践方法,使得测试者和开发者可以确保产品的可靠性达到用户要求、加快产品上市的速度、降低产品的成本、提高用户满意度以及降低产品风险。

软件可靠性工程的有效性,在于它运用了两个基本的思想[2]:第一,通过定量描述产品的操作方式,可以更有效地开发产品的功能并且使用这些信息将有限资源精确地集中到最

到稿日期:2008-10-28 返修日期:2009-01-04 本文受国家"863"项目(编号:2006AA01Z177),国家自然科学基金NSFC(编号:60873027),江苏省自然科学基金基础研究项目(编号:BK2006115)资助。

冯光成(1985-),男,硕士研究生,主要研究方向为软件工程与软件可靠性,E-mail:fgcmaster@gmail.com;顾 庆(1972-),男,教授,主要研究方向为分布式计算与并行处理;陈 翔(1981-),男,博士研究生,主要研究方向为测试用例生成与评价;陈道蓄(194-),男,教授,博士生导师,主要研究方向为分布式计算与并行处理等。

关键的功能上，使测试工作真实地反映实际条件；第二，软件可靠性工程平衡了用户对可靠性、开发时间和开发费用的需求。为此，软件可靠性工程要像对开发时间和开发费用设置定量目标那样，对可靠性也设置定量目标，并制定特定策略达到这些目标。最后，软件可靠性工程在测试过程中跟踪产品的可靠性，并用来作为产品是否可以发布的标准。通过软件可靠性工程，可以交付“正好合适”的可靠性产品，从而既避免了不必要的资金和时间浪费，又避免了发生由不够可靠的产品导致的用户不满意问题。

2.2 软件可靠性增长模型

软件可靠性增长模型（SRGM：Software Reliability Growth Models）是指在软件测试过程中，当失效被检测到则被随即移除。随着失效的排除，失效时间间隔会变长，软件的可靠性会增加，失效检测率会逐渐降低。当软件的失效检测率达到设定的目标要求后，就可以进行软件发布。软件可靠性增长模型是描述失效检测率的数学方程，大致可以分为两类[6]：凹模型（concave model）和S形模型（S-Shaped model），如图1所示。

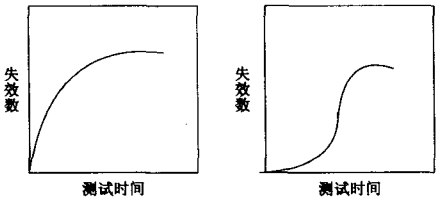

图1 凹模型和S形模型

凹模型的失效检测率随着测试时间的不断增加而逐渐下降，因此在图1中就呈现下凹形状。而S形模型则假设测试开始阶段的失效检测率没有测试后期阶段有效，因此在图1中呈S形状。凹模型主要包括Goel-Okumoto模型[7]、Musa-Basic模型[8]、Weibull模型[9]和Yamada Exponential模型[10]等，S形模型主要包括Gompertz模型[11]、Yamada Raleigh模型[12]、Delayed S-shaped模型[13]等。

3 软件可靠性增长模型选择框架（SaMS）

本文根据对以往工作的研究，提出了一种自适应的软件可靠性增长模型选择框架SaMS（Self-adaptive Model Selection for SRGM），它利用模型评价准则选择优选类模型，并利用神经网络方法进行可靠性的计算。方法对模型评价准则具有较好的扩展性，可根据项目实际需求设定不同的模型评价准则，对获得的可靠性增长模型利用实际数据进行训练，自适应地调整不同模型的权重。SaMS框架如图2所示。

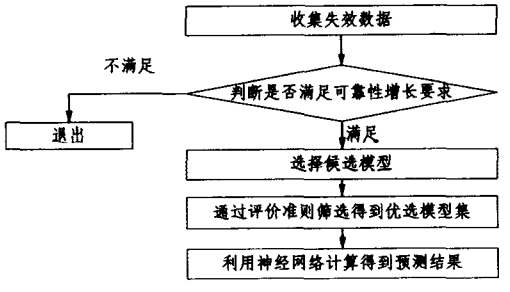

图2 软件可靠性增长模型选择与综合应用框架

SaMS：自适应软件可靠性增长模型选择框架

步骤1 收集软件失效数据，判断是否满足软件可靠性增长要求。在此步骤中，本文采用Laplace Test技术[14]来判断失效数据是否满足可靠性增长要求。

步骤2 利用失效数据，画出失效数据趋势图，根据趋势图选择候选的软件可靠性增长模型集。在选择过程中，通过分析软件失效检测机制以及失效报告过程，分析项目以及相关项目特性，结合模型假设条件获得候选模型集。

步骤3 对候选模型集进行拟合分析，应用不同模型评价准则对模型集进行排序，利用本文提出的选择算法筛选出优选模型集，并作为下一步的输入。

步骤4 根据上述步骤得到的优选模型集，以这些模型的预测结果作为输入，采用神经网络等技术进行综合，计算得到最终的预测结果。

框架对软件项目得到的失效数据进行可靠性相关指标的计算，可以利用模型综合的优点，避免了模型权重预先确定和利用单一模型来进行计算的不足。由于本框架重点在模型选择和模型综合两块，接下来就对模型选择和模型综合进行介绍，并提出本框架所使用的方法。

4 软件可靠性增长模型选择和综合

在本节中，首先介绍软件可靠性增长模型的选择，回顾当前模型选择的方法，然后提出一种对模型进行选择的优选模型选择算法，将它的结果作为模型综合的输入；介绍模型综合的相关知识，根据模型选择得到的优选模型利用神经网络对各种优选模型类进行权重的自动调整，最后计算得到可靠性。

4.1 基于评价准则的可靠性增长模型选择

传统的可靠性增长模型选择方法使用模型拟合性作为唯一的模型选择标准，通过选择模型拟合性最好的模型来计算可靠性。Woodcock等人提出利用赤池信息量准则（AIC：Akaike information criterion）作为选择模型的标准[15]，利用信息熵的观点来评估模型的准确性和复杂性。Stringefellow等人提出了一种经验的模型选择方法[16]，利用R^2作为拟合优度检验（GOF）指标，来评估曲线拟合的程度，并通过模型得到的两次预测数据之间的差值作为模型稳定性指标，并对R^2和模型稳定性设定阈值，在满足阈值条件的模型中再通过预测能力（预测值与实际值之差）来得到最优的模型。Carina Andersson将该经验方法应用到不同环境的软件项目中[17]，表明此法是可以复制的。Muhammad等人对实际应用中的30多种可靠性模型进行了分析[18]，并利用文中提到的9种标准在软件生命周期的不同阶段选择特定的模型。

本文利用Michael R. Lyu[4]提出的拟合优度检验（Good of Fit）、序列似然度（Prequential Likelihood）、模型噪音（Model Noise）以及Stringefellow[16]采用的R^2作为模型评价准则，通过对收集到的失效数据利用不同的评价准则对不同模型拟合计算，得到一系列模型评价指标。通过对这些指标进行分析，选定优选模型集作为下一步的输入。可供选择的软件可靠性增长模型包括J-M模型、M-B模型、NHPP模型、M-O模型、Delayed S-shaped模型和Gompertz模型等。通过失效数据计算得到各评价指标值，利用以下的选择算法得到的优选模型作为下一步的输入。

优选模型选择算法描述如下：

1）模型集合$M=\{m_1,m_2,\cdots,m_n\}$，可选模型个数为n，对候选模型$m_i\in M$，对于第j个评价指标得到p_{ij}；

2)根据项目进度以及可选模型个数，设定优选模型集规模 cnt；

3)对评价指标 j，按照模型表现的优劣得到排列$\langle m_1', m_2', \cdots, m_n'\rangle$，其中 m_i 优于 $m_j(i<j)$；

4)扫描得到排序结果，对于 j 评价指标的排列序列$\langle m_1', m_2', \cdots, m_n'\rangle$，利用智能 Agent 中的记分投票规则，$m_1$ 得 n 票，m_2 得 $n-1$ 票，$\cdots$，m_n 得 1 票，对每种评价指标重复此过程。对于 $m_i \in M$，计算该模型得到的总票数，取得票最多的前 cnt 个模型为候选模型；

4.2 基于神经网络的模型综合

单个软件可靠性增长模型虽然在其适用的子空间中可能会具有很好的效果，但适用范围有限。为了进一步提高预测的精度，可以将多种可靠性增长模型集综合起来预测可靠性。Michael R Lyu 通过对不同模型赋予不同的权重，提出了 4 种不同的可靠性综合模型[14]：ELC，MLC，ULC 和 DLC。这些都是通过现有的模型设置不同的权重来计算可靠性，此方法的缺点是权重需要预先确定且未考虑选择优选模型集。

本文根据尹乾等人在神经网络进行模型综合工作的基础上[19]，利用神经网络对优选模型集进行综合，根据优选模型得到的预测结果和实际结果训练神经网络，并动态地调节不同优选模型的权重。神经网络是由人工建立的以有向图为拓扑结构的动态系统，通过对连续或断续的输入做状态响应而进行信息处理。它将所有定量或定性的信息都等势分布贮存于网络内的各神经元，有很强的鲁棒性和容错性，可以充分逼近任意复杂的非线性关系，并具有自学习功能，往往可以获得较好的预测结果。在实际应用中，往往采用前馈神经网络(BP，Back-Propagation Network)和它的各种变体。BP 网络是一种对非线性可微分函数进行权值训练的多层网络，它的优点是仅仅借助样本数据，而无需建立系统的数学模型，就可对系统实现有 m 个输入神经元的模式向量 p 组成的 pm 空间到 yn 空间的高度非线性映射。BP 网络不仅具有输入层、输出层，而且有一个或多个隐藏点。输入的样本从输入层经过隐藏层进行处理，通过计算后，传向输出层。在输出层把实际输出和期望输出进行比较，如果实际输出不等于期望输出，则转向反向传播过程。反向传播时，把误差信号按原来正向传播的通路反向传回，并对每个隐层神经元权系数进行修改，以期误差信号趋向最小[20]。

在反向传播过程中，可利用下列步骤对网络的权系数 w_{ij} 进行递归求取。每层有 n 个神经元时，对于 k 层的第 i 个神经元，则有 n 个权系数$\{w_{i1}, w_{i2}, \cdots, w_{in}\}$，另外取 $w_{i,n+1}$ 用于表示阈值 θ_i。使用 BP 算法的过程如下：

1)对权系数 w_{ij} 置初值。对各层的权系数 w_{ij} 设置一个较小的非零随机数，并设置 $w_{i,n+1}=-\theta$。

2)输入一个样本 $x=(x_1, x_2, \cdots, x_n)$以及对应期望输出 $y=(y_1, y_2, \cdots, y_n)$。

3)计算各层的输出。对于第 k 层的第 i 个神经元的输出 x_i^k，有

$$x_i^k = f(U_i^k)$$

其中，$U_i^k=\sum_{j=1}^{n+1} w_{ij} x_j^{k-1}$，$x_{n+1}^{k-1}=1$，$w_{i,n+1}=-\theta$，$f$ 为激发函数。

4)求各层的学习误差 d_i^k。对于输出层 $k=m$，有 $d_i^m=x_i^m(1-x_i^m)(x_i^m-y_i)$；对于其他各层，有 $d_i^k=x_i^k(1-x_i^k)\sum_l W_{li} d_j^{k+1}$。

5)修正权系数与阈值。

$$w_{ij}(t+1)=w_{ij}(t)-\eta \cdot d_i^k \cdot x_j^{k-1}$$

其中，η 为学习速率，即步长，η 取值区间为[0.1，0.4]。

6)当求出各层各个权系数之后，可按给定品质指标判别是否满足要求。如果满足要求，则算法结束；如果未满足要求，则返回 3)执行。

5 实例分析

本文中使用的数据来自文献[5]，所使用的 SYS3 和 CSR1 是在实际 NASA 软件项目中获取的软件可靠性失效数据。SYS3 数据集中共有 207 组采样点，每组采样点包含失效数和对应的执行时间。取前面 75%数据用来进行模型的选择，剩余的数据用作测试。根据上述步骤选择 Gompertz 模型、J-M 模型、Delayed S-shaped(DS)模型、M-B 模型、M-O 模型和 NHPP 模型作为候选模型集，计算得到表 1 的各模型评价指标值。

表 1 SYS3 数据集各种模型评价指标值

	GOF	PL	Model Noise	R^2
Gompertz	0.07695	582.55	3.3234	0.875
J-M	0.06013	582.46	1.9813	0.915
DS	0.07043	582.38	3.2803	0.892
M-B	0.05805	582.21	2.0945	0.918
M-O	0.06969	582.14	1.6744	0.901
NHPP	0.07903	582.78	1.6933	0.883

根据本文的选择算法，选择 J-M，M-B 和 M-O 模型作为模型综合的输入，通过不断的训练，获得预测的失效数和对应的执行时间。将 3 种模型得到的预测值和利用 SaMS 方法得到的预测值进行归一化处理，做预测图，得到图 3。

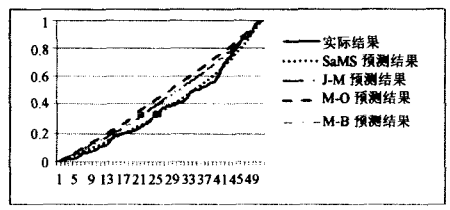

图 3 SYS3 失效数据集预测结果

利用 Michael Lyu 提到的均方根误差(RMSE)方法[11]来比较 3 种单一模型和训练得到的神经网络的预测结果。

$$RMSE=\frac{1}{n-1}\sqrt{\frac{\sum_{k=1}^{n}(C(k)-\hat{C}(k))^2}{\sum_{k=1}^{n}(C(k))^2}}$$

其中，$C(k)$是 k 时刻发生的错误数，$\hat{C}(k)$是 k 时刻预测的错误数。RMSE 取值越小，表明预测结果越好。

计算得到的结果如表 2 所列。

表 2 各模型的 RMSE 值

模型	J-M	M-O	M-B	SaMS
RMSE	0.0469	0.0514	0.0465	0.00292

CSR1 数据集中共有 398 组采样点，每组采样点包含失效数和对应的执行时间。取前面 75%的数据用来进行模型的选择以及神经网络的训练，剩余的数据用作测试。根据上述步骤选择 Gompertz 模型、J-M 模型、Delayed S-shaped 模型、M-B 模型、M-O 模型和 NHPP 模型作为候选模型集，计算得到表 3 所列的各模型评价指标值。

表 3 CSR1 数据集各种模型评价指标值

	GOF	PL	Model Noise	R^2
Gompertz	0.4639	751.69	4.1487	0.904
J-M	0.2134	759.66	1.2416	0.954
DS	0.2025	793.42	3.9195	0.893
M-B	0.2019	747.68	0.8856	0.976
M-O	0.2419	788.87	3.0287	0.915
NHPP	0.1915	768.05	1.5912	0.981

选择 J-M, M-B 和 NHPP 模型作为模型综合的输入，通过不断的训练，获得预测结果。将 3 种模型得到的预测值和利用 SaMS 方法得到的预测值进行归一化处理，做预测图，得到图 4。

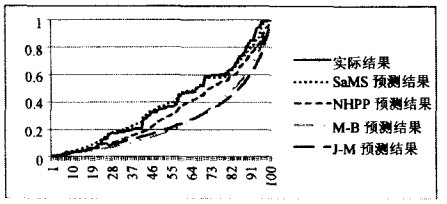

图 4　CSR1 失效数据集预测结果

同样利用 RMSE 方法来比较 3 种单一模型和 SaMS 的预测结果，如表 4 所列。

表 4　各模型的 RMSE 值

模型	J-M	MB	NHPP	SaMS
RMSE	0.0574	0.0586	0.0398	0.00247

根据以上的预测图可以直观地看出，利用本文提出的方法得到的拟合效果是最好的。通过 RMSE 值也说明了本文方法比单一优选模型具有更好的效果。本文利用模型评价准则来评估各种不同模型对失效数据的预测能力，选择优选模型集作为神经网络的输入，而不仅仅依靠模型评价准则选择单一模型得到预测结果；利用优选模型集良好预测能力来训练人工神经网络，克服了较差模型对神经网络训练的干扰，既可以利用神经网络较好的自学习功能，又避免了对优选模型权重的确定。

结束语　本文总结了软件可靠性模型目前常有的一些选择和综合方法，提出了一个自适应的可靠性增长模型选择和综合框架 SaMS，利用实际获得的软件失效数据并通过实验结果验证了它的有效性。目前的工作包括对模型选择问题中评价准则的研究，选择增强型的神经网络方法对模型进行更好的综合。

参 考 文 献

[1] ANSI/IEEE. Standard Glossary of Software Engineering Terminology[S]. ANSI/IEEE: STD-729-1991, 1991

[2] Musa J D. Software Reliability Engineering (V2) [M]. Author House, 2004

[3] Jelinski M Z. Software reliability research[C]// Statistical Computer Performance Evaluation. New York: Academic Press, 1972: 465-484

[4] Cai K, Wen C, Zhang M. A critical review on software reliability modeling [J]. Reliability Engineering and System Safety, 1991: 357-371

[5] Lyu M R. Handbook of Software Reliability Engineering [M]. New York: IEEE Computer Society Press, McGraw Hill, 1996

[6] Wood A. Predicting Software Reliability [J]. IEEE Computer, 1996, 29(11): 69-77

[7] Goel A L, Okumoto K. A Time Dependent Error Detection Model for Software Reliability and Other Performance Measures [J]. IEEE Transactions on Reliability, 1979, 28(3): 206-211

[8] Musa J D. A theory of software reliability and its application [J]. IEEE Transactions on Software Engineer, 1975, 1(3): 312-327

[9] Ascher H, Feingold H. Repairable System Reliability: Modeling, Inference, Misconceptions, and their Causes [M]. Marcel Dekker Inc, 1984: 103-111

[10] Yamada S, Ohtera H, Narihisa H. Software Reliability Growth Models with Testing Effort[J]. IEEE Transactions on Reliability, 1986, 35(1): 19-23

[11] Kececioglu D. Reliability Engineering Handbook [M]. NJ: Prentice-Hall, 1991

[12] Yamada S. Software quality reliability measurement and assessment: Software reliability growth models and data analysis [J]. Journal of Information Processing, 1991, 14(3): 254-266

[13] Yamada S, Ohba M, Osaki S. S-shaped reliability growth modeling for software error detection [J]. IEEE Transactions on Reliability, 1983, 32(5): 475-478

[14] Lyu M R, Nikora A. CASRE: A Computer-Aided Software Reliability Estimation Tool [J]. Computer-Aided Software Engineering, 1992: 264-275

[15] Khoshgoftaar T M, Woodcock T G. Software Reliability Model Selection: A Case Study [J]. ISSRE, 1991: 183-191

[16] Stringefellow C, Andrews A. An Empirical Method for Selecting Software Reliability Growth Models [J]. Empirical Software Engineering, 2002, 7(4): 319- 343

[17] Andersson C. A replicated empirical study of a selection method for software reliability growth models [J]. Empirical Software Engineering, 2007, 12(8): 161- 182

[18] Asad C A, Ullah M I. An Approach for software reliability model selection [C]// Proceedings of the 28th Annual International Computer Software and Applications Conference. COMPSAC'04. Sep. 2004

[19] Yin Qian, Li Jiang. A New Cascade Software Reliability Model [C]// Third International Conference on Natural Computation. ICNC'2007. Aug. 2007: 241-245

[20] Su Yu-Shen, Huang Chin-Yu, et al. An Artificial Neural-Network-Based Approach to Software Reliability Assessment [C]// TENCON'2005. Nov. 2005

(上接第 78 页)

[5] Adnan M, Alhajj R, Barker K. Performance analysis of incremental update of association rules mining approaches[M]. Intelligent Engineering Systems. Mediterranean Sea, 2005

[6] 朱玉全，孙志挥，季小俊. 基于频繁模式树的关联规则增量式更新算法[J]. 计算机学报，2003，26(1)：91-96

[7] Lin Ming-Yen, Lee Suh-Yin. Incremental update on sequential patterns in large databases by implicit merging and efficient counting[J]. Information Systems, 2004, 29(5): 385-404

[8] 易彤，徐宝文，吴方君. 一种基于 FP 树的挖掘关联规则的增量更新算法[J]. 计算机学报，2004，27(5)：703-710

[9] Seong - Ryong K, Loguinov D. Modeling Best - Effort and FEC Streaming of Scalable Video in Lossy Network Channels[J]. IEEE/ACM Transactions, 2007, 15(1): 187-200

[10] Law K L E, Saxena A. Scalable Design of a Policy-Based Management System and its Performance[J]. IEEE Communications, 2003, 41(6): 72-79