doi:10.11918/j.issn.0367-6234.2015.05.006

SRGM 中 TE 建模机制与模型比较分析

张 策[1,2], 孟凡超[2], 崔 刚[1], 刘宏伟[1], 官红玉[2]

(1.哈尔滨工业大学 计算机科学与技术学院,150001 哈尔滨;
2.哈尔滨工业大学(威海) 计算机科学与技术学院,264209 山东 威海)

摘 要: 针对当前与软件可靠性增长模型 SRGM(software reliability growth model)紧密相关的测试工作量 TE(testing effort)的研究现状,以及全面评价测试工作量函数 TEFs(testing effort function)性能的需要,对 SRGM 下 TE 相关的建模与模型比较问题进行深入研究. 分析了近 40 年的 TE 发展历程与内容演变,对其与测试成本间关联进行剖析;对典型的 TEFs 进行了全面梳理与述评;研究了考虑 TE 的 SRGM 建模过程;针对 8 个典型的 TEFs,在已发表的 4 组数据集上进行了深入的性能比较与评述. 研究结果表明,有效的 TEF 可为基于 SRGM 的可靠性相关研究提供重要决策支持,对考虑 TE 的 SRGM 进行综合评价是后续研究的关键.

关键词: 软件可靠性;测试工作量;测试工作量函数;软件可靠性增长模型;成本
中图分类号: TP311 **文献标志码**: A **文章编号**: 0367-6234(2015)05-0032-08

Overview of modeling of TE in SRGM and comparisons for models

ZHANG Ce[1,2], MENG Fanchao[2], CUI Gang[1], LIU Hongwei[1], GUAN Hongyu[2]

(1. School of Computer Science and Technology, Harbin Institute of Technology, 150001 Harbin, China;
2. School of Computer Science and Technology, Harbin Institute of Technology at Weihai, 264209 Weihai, China)

Abstract: To systematically reveal the state of art of testing effort (TE) related to software reliability growth model (SRGM) and to evaluate performances of testing effort function (TEFs), modelling of TE dependent SRGM and comparisons for models are studied in this paper. More specifically, firstly, research evolution of TE is summarized since the early 1978 s, and the relation between TE and test cost is analyzed. Then, current TEFs are reviewed and discussed, and modeling process of SRGM incorporating TE is illustrated. Performances of 8 TEFs are evaluated and compared by 4 failure data sets published. Finally, conclusions indicate that appropriate TEF can support decision for research of reliability based on SRGM and evaluating SRGM incorporating TE would be worthwhile in future.

Keywords: software reliability; testing effort (TE); testing effort function (TEF); software reliability growth model (SRGM); cost

软件测试中,在测试工作量 TE(testing effort)的不断消耗下,故障不断被检测和排除掉,总故障数量 $a(t)$ 保持下降趋势,使得软件可靠性 $R(t)$ 获得增长. TE 的消耗代表着测试成本的支出,而软件测试通常受制于有限的预算限制;同时,TE 的消耗进度也应该受到有效的控制. 以可靠性为目标之一的测试过程应被合理的管控,低效与冗余的测试可能导致额外的成本开销[1],对成本管控和最优发布时间决策带来隐患. TE 对以软件可靠性增长模型 SRGM(software reliability growth model)为基础的可靠性工程相关研究形成了重要支撑.

本文对 TE 的建模过程,考虑 TE 的 SRGM 建模,以及现有的测试工作量函数 TEF(testing effort

收稿日期: 2014-06-03.
基金项目: 国家科技支撑计划资助(2013BA17F02);山东省科技攻关项目(2011GGX10108, 2010GGX10104).
作者简介: 张 策(1978—),男,讲师;博士研究生;
崔 刚(1949—),男,教授,博士生导师;
刘宏伟(1971—),男,教授,博士生导师.
通信作者: 张 策, zhangce@hitwh.edu.cn.

function)评价进行了综合研究与评述. 首先给出了有关概念,对 TE 的发展历程与研究内涵进行分析,并对当前 TEFs 进行了全面梳理;然后对考虑 TE 的 SRGM 建模机制进行剖析;最后在 4 个已发表的数据集上对 8 个典型的 TEFs 进行了全面比较.

1　定义与相关背景

1.1　相关定义

定义 1　测试工作量 TE[2-4]用以描述软件测试过程中所消耗的资源,可表示为测试中所花费的测试案例与分析数据的数量、CPU 时间、时钟时间、日历时间数或者人力费等.

定义 2　测试工作量函数 TEF 是采用数学表达式的形式对测试过程中 TE 的消耗进行描述的数学工具.

定义 3　软件可靠性增长模型 SRGM[5-6]是在软件开发的测试阶段与操作运行阶段,基于失效数据,描述软件测试过程中累积检测或修复的故障数量,TE 与测试时间等的数学关系,是实现建模软件可靠性提高过程的数学工具.

1.2　TE 发展历程变迁

与 TE 紧密相关的研究最早可追溯到文献[7],近 40 年来科研人员对 TE 的研究伴随着 SRGM 的广泛应用得到了足够重视,图 1 给出了 TE 的发展历程及研究内容的演变. 可以看出,自 20 世纪 90 年代后期始,TE 与 SRGM 开始融合,并在以可靠性为核心的研究中占有重要地位.

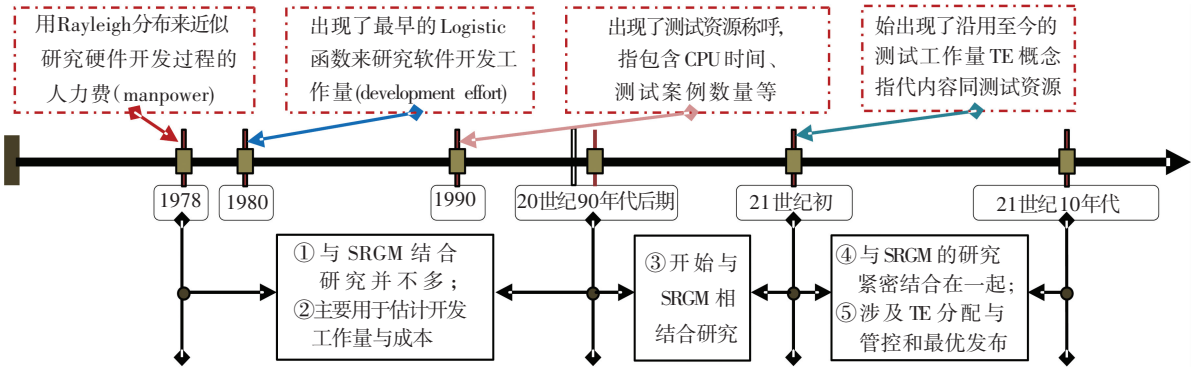

图 1　TE 发展历程与研究内容演变

与 SRGM 相关的基于 TE 的研究主要包括:1)考虑 TE 进行 SRGM 的建模,提高模型的有效性;2)进行 TE 的分配与管控,实现测试过程中对以可靠性为核心的指标调整;3)建立成本模型,在“成本+可靠性”的综合统筹下,实现软件的预期发布.

1.3　讨论:TE 与测试成本的区别与关系

在 TE 与测试成本的关系上,二者紧密相关但存有明显区别:TE 的本质是如其定义中所述的测试资源;测试成本指测试过程中所花费的总成本 $C(T)$. 虽有文献[8-9]认为 TE 就是测试成本,但显然是片面的,因为只是全部软件测试成本中包含了 TE 花销的成本而已[10]. 例如,测试成本在文献[1, 5]中被采用

$$C(T) = C_0 + C_1 m(T) + C_2 [m(T_{LC}) - m(T)] + C_3 \int_0^T w(t)\,\mathrm{d}t. \tag{1}$$

式中:T_{LC} 为软件生命周期长度;C_0 为初值;C_1、C_2 分别为测试与运行阶段改正一个故障的单位成本,满足 C_2 至少一阶大于 C_1;C_3 为单位 TE 花销的成本,而全部 TE 花销的成本为 $C_3 \int_0^T w(t)\,\mathrm{d}t$.

2　TEFs 类别与讨论

软件测试中,TE 被不断消耗的过程表示了失效被如何有效地检测到[11]. TEF 能被用来描述测试过程中的工作剖面,是对(测试)资源消耗的一种定量描述,存在多种形式.

2.1　Yamada Exponential、Yamada Rayleigh 与 Yamada Weibull 型 TEF:基于微分方程法

基于微分方程法来建立并求解 TEF 通常是基于以下假设[2,11-12]:t 时刻测试工作量消耗率与当前剩余可得的测试工作量成正比,比例函数为 $\tilde{\omega}(t)$. 这样,可得到微分方程

$$\frac{\mathrm{d}W(t)}{\mathrm{d}t} = w(t) = \tilde{\omega}(t)\,[W - W(t)].$$

式中:W 为全部可得的测试工作量;$W(t)$ 为 TEF;$\tilde{\omega}(t)$ 为 t 时刻当前剩余测试工作量相关的率函数. 在 $W(0) = 0$ 的初始条件下,求解可得到

$$W(t) = W[1 - \mathrm{e}^{\int_0^t \tilde{\omega}(\tau)\,\mathrm{d}\tau}].$$

1)当 $\tilde{\omega}(t) = \beta$ 常量时,可以得到 Yamada Exponential(指数)型 TEF 为

$$W(t) = W[1 - \mathrm{e}^{-\beta t}]. \tag{2}$$

2) 当 $\widetilde{\omega}(t) = \beta t$ 线性时,可以得到 Yamada Rayleigh(瑞利)型 TEF 为

$$W(t) = W[1 - e^{-(\frac{\beta t^2}{2})}]. \tag{3}$$

3)更一般地,当 $\widetilde{\omega}(t) = \beta\delta t^{(\delta-1)}$ 指数时,可以得到 Yamada Weibull(威布尔)型 TEF 为

$$W(t) = W[1 - e^{-\beta t^{\delta}}]. \tag{4}$$

显然,式(2)与式(3)是式(4)的一种特殊情况.

2.2 Exponential Weibull 型 TEF

在文献[13]的基础上,文献[14-15]提出了一般化的 EW TE(Exponentiated Weibull:指数威布尔型)对软件失效行为进行预测,并认为 EW TEF 在评估可靠性过程中表现优异,有较强的适应性与柔韧性.

$$W(t) = W(1 - e^{(-\beta\cdot t^{\delta})})^{\theta},$$
$$W > 0, \beta > 0, \delta > 0, \theta > 0. \tag{5}$$

式中:β 为比例调节参数,δ、θ 分别为形状参数.对此 3 个参数进行讨论,得到表 1 中结果.

表 1 EW TEF 特定条件下演化结果

条件	类型	TEF 表达式
$\theta = 1, \delta = 1$	Exponential 型 TEF	式(2)
$\theta = 1, \delta = 2$	Rayleigh 型 TEF	式(3)
$\theta = 1$	Weibull 型 TEF	式(4)
$\delta = 1$	一般的 Exponential 型 TEF	$W(t) = W(1 - e^{(-\beta t)})^{\theta}$
$\delta = 2$	一般的 Burr 型 X TEF	$W(t) = W(1 - e^{(-\beta t^2)})^{\theta}$

2.3 泛 Logistic 型 TEF

作为 Rayleigh TEF 的替代,Logistic TEF 最早由 Parr[16] 于 1980 年提出,后被广泛研究. 文献[17]认为 Weibull 类型的 TEF 当 $\delta > 3$ 时会出现所谓的"峰现象",并不符合实际软件测试,据此文献[1,3,5,10,17-18]提出了 Logistic TEF、一般化的 Logistic TEF 和更一般化的 Logistic TEF,分别为

$$W_k(t) = \frac{W}{1 + Ae^{(-\alpha t)}}. \tag{6}$$

$$W_{gk}(t) = \frac{W}{\sqrt[k]{1 + Ae^{(-\alpha k t)}}}. \tag{7}$$

$$W_{k\beta}(t) = \frac{W\sqrt[k]{\frac{k+1}{\beta}}}{\sqrt[k]{1 + Ae^{(-\alpha k t)}}}. \tag{8}$$

此外,在文献[19]中提出 Log-Logistic TEF 为

$$W(t) = \left[\frac{(\theta t)^{\delta}}{1 + (\theta t)^{\delta}}\right] W. \tag{9}$$

整体而言,由于 Logistic TEF 具有 S 型的变化趋势,可描述多种 TE 消耗情形,得到了广泛应用.

2.4 讨论:TEFs 必要说明

显然,随着测试的进行,TE 的花销应呈现不断累积增大的趋势. 对表 1 中的 TEF 进行求导,可以得到 $\frac{dW(t)}{dt} > 0$, 这样其满足上述要求. 例如

$$\frac{dW_k(t)}{dt} = w_k(t) = \left[\frac{A\alpha k e^{(\alpha k t)}}{(1 + Ae^{(\alpha k t)})^2}\right] W > 0.$$

在测试开始时,由于准备工作等的存在使得需要消耗部分 TE,因而使得 TEF 的初值不等于 0,即 $W(t) \neq 0$. 例如, $W_k(0) = \frac{W}{\sqrt[k]{1 + A}} \neq 0$, $W_{k,\beta}(0) = \left(\sqrt[k]{\frac{k+1}{\beta}} / \sqrt[k]{1 + A}\right) W \neq 0$. 这在确定成本构成与最优发布问题的研究中具有重要意义:在式(1)中,很多研究均是人为设定 C_0 的初值,具有很强的不确定性,本文认为 $C_0 = f(W(0))$[20],很好的解决了初值设定问题.

现有 TEF 模型在数学表达式上存在很大差异,因而其在不同的数据集的表现上一定会存在较大不同.

3 考虑 TEF 的 SRGM 建模研究

在 SRGM 的相关研究中,对 TE 的研究均是借助 TEF 来进行的,主要涵盖 SRGM 建模、TE 管控和最优发布 3 个方面上,且前者是后两者研究的基础. TE 为 3 个问题的研究提供了必要的支撑,使得所建立的 SRGM 更加考虑到实际测试因素,对 TE 的管控更为高效,最优发布具备决策参考支持,并最终为获得预期的可靠性目标提供重要保障. 在文献[6]中,已对 TE 分配与调控和最优发布策略进行分析,这里从略.

3.1 考虑 TEF 的 SRGM 基本建模方法

测试阶段软件可靠性与故障检测与修复所花费的 TE 紧密相关[11],通常所建立的 SRGM 中直接包含有 TEF. 例如,典型的考虑 TE 到 SRGM 中的建模方法如下[1-3,5,10-11,15,17-18,21-25]:

$$\frac{dm(t)}{dt} \cdot \frac{1}{w(t)} = r(t) \cdot [a - m(t)].$$

式中：$m(t)$ 为$[0, t]$ 内累计检测到的故障数量，a 为软件中的总故障个数. 该微分方程建立的假设基础为：$[0, t]$ 内累计检测到的故障数量与当前剩余的故障数量下所花费的 TE 时的故障检测率 $r(t)$ 成比例.

3.2　讨论：不完美排错下的建模

由于测试环境中随机因素的影响，考虑 TEF 的 SRGM 应向不完美排错研究[12, 25]的趋势转变，将更多的实际因素融入进来. 为此，文献[20, 26]给出了更加一般化且具有较强柔韧性的框架模型，支持不完美排错与 TE 的综合考量为

$$\begin{cases}\dfrac{\mathrm{d}m(t)}{\mathrm{d}t}\cdot\dfrac{1}{w(t)}=r(t)\cdot(a(t)-c(t)),\\ \dfrac{\mathrm{d}a(t)}{\mathrm{d}t}=h(t)\cdot\dfrac{\mathrm{d}c(t)}{\mathrm{d}t},\\ \dfrac{\mathrm{d}c(t)}{\mathrm{d}t}=p(t)\cdot\dfrac{\mathrm{d}m(t)}{\mathrm{d}t}.\end{cases}\tag{10}$$

式中：$a(t)$、$c(t)$ 分别为$[0, t]$ 内软件中总的故障数量和累计修复的故障数量；$h(t)$、$p(t)$ 分别为故障引入比例函数和故障改正比例函数. 式(10) 描述了更多的测试细节，可以演变为多个现有的模型；同时，在现有研究的基础上给出了改进的初值不为零的 Logistic TEF 为

$$W(t)=\left(\frac{1-v\mathrm{e}^{-\alpha t}}{1+u\mathrm{e}^{-\alpha t}}\right)W.\tag{11}$$

基于此所获得的 SRGM 在性能上优于其他模型[20].

由于 TEF 的引入，测试过程中可以借助调整 TE 的方式来影响 $m(t)$，使得测试中的可靠性能向预期增长(测试阶段的可靠性)为

$$R(x\mid T)=\mathrm{e}^{-[m(T+x)-m(T)]}.$$

相比于传统 SRGM，考虑 TEF 的 SRGM 表现得更为优秀，这可以被解释为 TEF 的偏差在它的基本概念中已经考虑到了人力因素和不确定性.

软件发布不仅要考虑到可靠性要求，成本因素也至关重要[18]，“成本+可靠性”综合标准[15]已成为最优发布的统筹考虑.

4　TEF 性能验证与分析

4.1　性能评价标准与数据集选择

在对模型与真实数据的拟合评价上，MSE，MEOP，Variation，RMS-PE，TS 和 BMMRE 常被用来作为度量的选择，这些数值越小表明拟合效果越好，R-square 越接近于 1 越好；RE 被作为模型对未来数据预测的评价标准，RE 越趋近于 0 表明预测效果越好.

$$\mathrm{MSE}=\frac{1}{K}\sum_{i=1}^{K}(W(t_i)-W_i)^2.$$

$$\mathrm{MEOP}=\sum_{i=k}^{K}|W(t_i)-W_i|/(K-p+1).$$

$$\mathrm{Variation}=\sqrt{\sum_{i=1}^{K}\frac{((W_i-W(t_i))-\mathrm{Bias})^2}{(K-1)}}.$$

$$\mathrm{RMS-PE}=\sqrt{\mathrm{Variation}^2+\mathrm{Bias}^2}.$$

$$\mathrm{Bias}=\frac{1}{K}\sum_{i=1}^{K}(W_i-W(t_i)).$$

$$\mathrm{TS}=\sqrt{\sum_{i=1}^{K}(W(t_i)-W_i)^2/\sum_{i=1}^{K}W_i^2}\times 100\%.$$

$$R\text{-square}=\frac{\sum_{i=1}^{K}[W(t_i)-\bar{W}]^2}{\sum_{i=1}^{K}[W_i-\bar{W}]^2},\bar{W}=\frac{1}{K}\sum_{i=1}^{K}W_i.$$

$$\mathrm{BMMRE}=\frac{1}{K}\cdot\sum_{i=1}^{K}\left|\frac{\mathrm{PE}_i}{\min(W(t_i),W_i)}\right|.$$

$$\mathrm{RE}=\frac{W(t_q)-q}{t_q}.$$

式中：K 为数据样本数量大小；p 为参数个数；W_i 为到 t_i 时实际观测到的数据；$W(t_i)$ 为到 t_i 时利用模型得到的估算值.

关于 TE 数据集，本文选择由世界知名公司或机构发布的且已被广泛地应用的 TE 数据集：DS_1[27]，DS_2[28]，DS_3[29]和 DS_4[30]来作为验证的对象，如表 2 所示. 需要指出，由于不同公司在收集与记录测试资源消耗的方式上的不同，这 4 组数据集中的数据值差异较大.

4.2　参与比较的 TEFs

这里，本文选取典型的 8 个 TEFs 模型用于比较，如表 3 所示，并依次编号为①～⑧. 其中③～⑤为由⑥中的模型演化而来.

4.3　TEF 性能差异化比较

首先，对表 3 中的 TEFs 在 DS_1～DS_4上进行参数拟合，并画出了 TEFs 在 4 组数据集上的实际 TE 消耗曲线与 $W(t)$ 之间的拟合情况，如图 2 所示.

1) 在 DS_1、DS_2和 DS_4上 8 个 TEFs 与实际的 TE 消耗情况拟合的较好，而且 $W(t)$ 曲线基本呈现出通过原点的斜率为 k 的直线态势. 例如，从图 2 的 4 幅图中均可以看出，$W(t)=W(1-\mathrm{e}^{-\varphi t})$ 的曲线接近于通过原点的线性形状. 造成这种现象的原因以及斜率 k 的近似计算可解释为：对于这些数据集其估计得到的参数值 $\hat{\varphi}$ 非常的小(例如，DS_1中，$\hat{\varphi}=0.000\,590\,895\,2$). 此时，可以得到 $\lim\limits_{\varphi\to 0}W(t)=\lim\limits_{\varphi\to 0}W[1-\mathrm{e}^{-\varphi t}]=W\varphi t$，因而呈现线性关系，其斜率为 $k=W\varphi$.

表 2 TE 数据集遴选—真正的实际应用案例

TE 实例	概要分析
TE-DS_1[27]	由 IBM 公司发布,来自 IBM 公司开发的数据库应用软件:PL/I 应用系统的测试过程. PL/I 应用系统由 1 317 000 条代码构成,以周为单位进行记录累积失效个数和累积花费的 TE,合计记录了 19 周. 该数据集中指出 TE 的单位是执行时间,但并没有明确说明该执行时间是否是 CPU 花费的. 从第 1 周记录的 2.45 一直持续到第 19 周记录的 TE 数值:46.65.
TE-DS_2[28]	由国际著名的容错计算机公司 Tandem(天腾)发布的 TE 数据集,来自天腾公司一个计算机工程项目,以周作为记录单位,合计观测了 20 周累积失效个数和 TE 数据. TE 数据采用 CPU 执行时间作为测试资源消耗的记录单位,TE 值从 519 持续记录到 10 000.
TE-DS_3[29]	来自于罗马航空发展中心 RADC(Rome Air Development Center)一个项目:T1 系统发布的 TE 数据集,T1 系统由贝尔实验室(Bell Laboratories)的 9 名工程师开发,包含 21 700 条指令,被用于实时命令与控制应用. 其同样以周作为记录单位,合计记录了 21 周的数据,包括累积检测的失效个数、累积修复的失效个数和累积测试工作量 TE 数据. TE 数据也是以 CPU 执行时间作为单位,TE 值从 0.009 17 持续记录到 25.300 17.
TE-DS_4[30]	该数据集来源于一个在线数据采集软件包,其软件大小为 40 000 行代码. 其失效数据和 TE 的观测以天为单位,从第 1 天观测到第 21 天.

表 3 参与比较的 TEFs

模型	$W(t)$ 形式	模型	$W(t)$ 形式
Logistic	①: 式(6)	Yamada Weibull	⑤: 式(4)
Generalized Logistic	②: 式(7)	Exponential Weibull	⑥: 式(5)
Yamada Exponential	③: 式(2)	Log-Logistic	⑦: 式(9)
Yamada Rayleigh	④: 式(3)	Deformable Logistic	⑧: 式(11)

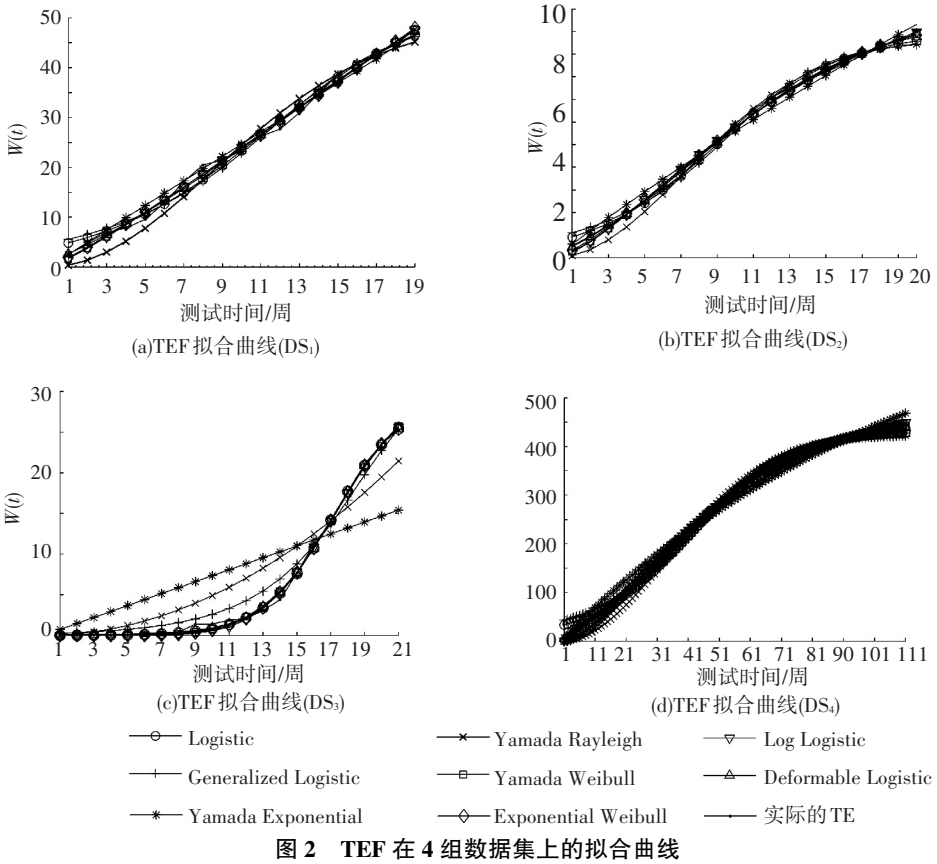

图 2 TEF 在 4 组数据集上的拟合曲线

2)在 DS_3 上,TEFs 的拟合差别比较明显,其中②、③和④出现了较大的偏差,其余模型拟合效果较好. 这主要是由于 DS_3 中实际的 TE 数据与②、③和④模型函数的本质差距较大造成的.

为了进一步区分 TEFs 在数据集上的拟合差异,本文计算了具体的比较标准值,结果如表 4 所示.

为了清晰地显示出 TEFs 在 4 组数据集上不同标准上的差异,表 4 中每个数据集上,性能值最优的采用 * 的形式标出,其次采用双下划线的形式标出,再次采用单下划线的形式标出. 由表 4 可以明显看出:

表 4　TEFs 性能标准比较结果

标准		模型							
		①	②	③	④	⑤	⑥	⑦	⑧
MSE	DS_1	1.627 200	2.287 567	1.660 643	5.147 693	0.897 335	0.858 227	0.904 443	0.821 816 *
	DS_2	37 786.668 653	82 156.984 115	91 673.250 894	113 337.048 757	19 248.779 403	17 031.677 833	22 820.167 646	13 057.920 757 *
	DS_3	0.120 235	1.432 146	27.880 067	9.084 125	0.177 528	0.166 349	0.194 446	0.114 479 *
	DS_4	92.276 309	203.595 764	281.901 318	208.438 693	29.905 339	25.462 470	47.751 189	19.394 269 *
MEOP	DS_1	1.092 982	1.333 581	1.109 525	2.041 612	0.891 724	0.934 847	0.893 915	0.836 435 *
	DS_2	174.631 319	265.572 596	282.498 839	285.336 939	127.841 133	137.373 164	134.076 985	113.692 881 *
	DS_3	0.242 218 *	1.080 989	4.885 012	2.633 013	0.321 811	0.303 860	0.334 587	0.281 055
	DS_4	7.878 673	12.059 922	15.256 387	11.984 488	4.102 802	4.146 095	5.493 261	3.547 237 *
Variation	DS_1	1.306 680	1.547 071	1.273 103	2.169 302	0.970 445	0.950 968	0.974 074	0.931 376 *
	DS_2	198.443 520	292.144 406	305.521 060	322.079 209	141.100 461	133.461 783	153.559 101	117.239 638 *
	DS_3	0.350 832	1.040 326	5.147 569	2.815 108	0.428 006	0.405 003	0.431 670	0.346 703 *
	DS_4	9.580 219	14.219 238	16.553 915	13.675 182	5.449 583	5.056 977	6.863 598	4.423 865 *
RMS-PE	DS_1	1.310 368	1.553 556	1.321 343	2.322 794	0.973 088	0.951 748	0.976 924	0.931 382 *
	DS_2	199.388 215	293.980 275	310.387 511	344.272 856	142.282 277	133.874 121	154.916 712	117.239 829 *
	DS_3	0.355 100	1.218 066	5.398 319	3.075 955	0.431 568	0.417 324	0.450 910	0.346 703 *
	DS_4	9.648 998	14.332 385	16.863 276	14.495 636	5.492 985	5.068 812	6.940 864	4.423 865 *
R-square	DS_1	0.968 030	0.954 158	0.905 598	1.175 710	1.012 435	1.005 080	1.013 054	0.995 543 *
	DS_2	0.972 673	0.951 710	0.924 230	1.134 500	1.019 262	1.010 837	1.022 234	0.998 378 *
	DS_3	1.008 250	0.864 981	0.314 218	0.646 547	1.014 459	1.016 141	1.020 862	0.998 437 *
	DS_4	0.964 143	0.940 973	0.898 557	1.119 978	1.017 197	1.008 113	1.025 624	0.999 026 *
TS	DS_1	0.045 429	0.053 864	0.045 893	0.080 801	0.033 736	0.032 992	0.033 869	0.032 285 *
	DS_2	0.030 245	0.044 597	0.047 109	0.052 381	0.021 587	0.020 306	0.023 504	0.017 780 *
	DS_3	0.032 626	0.112 602	0.496 820	0.283 592	0.039 645	0.038 376	0.041 491	0.031 836 *
	DS_4	0.031 573	0.046 898	0.055 185	0.047 453	0.017 974	0.016 585	0.022 713	0.014 475 *
BMMRE	DS_1	0.106 693	0.140 262	0.064 076	0.637 421	0.083 334	0.070 643	0.084 933	0.047 254 *
	DS_2	0.071 296	0.113 505	0.086 912	0.410 858	0.084 715	0.060 521	0.100 539	0.022 666 *
	DS_3	0.243 726 *	5.288 730	23.067 607	5.718 521	570.484 412	2 242.157 281	11 188.730 600	1.425 397
	DS_4	0.181 814	0.259 411	0.129 061	0.591 066	0.101 487	0.058 112	0.182 102	0.019 833 *

1）8 个 TEFs 在 4 组数据集上的比较结果中，第⑧个 TEF 模型（变形 Logistic 型）的结果值是最优的：其在 28（4 * 7）个性能值中有 26 个数值是最优的，且另外 2 个是次优的.

2）其次是第⑥个 TEF 模型（一般指数型），其有 17 个在剩余 7 个 TEF 模型的比较标准中最优，另有 7 个标准是次优的.

3）再次是第⑤个 TEF 模型（Yamada Weibull 型），在除了第⑧个模型外剩余的 6 个模型的比较标准中，有 19 个是最优的.

因而，本文可以充分地认为，第⑧个 TEF 个可以最好的描述多种情况的 TE 消耗，这可以解释为变形 Logistic 型具有先增后降的 S 型变化趋势，符合多数测试工作量的实际消耗情况.

接下来，本文画出了 TEFs 在 4 个数据集上的预测曲线 RE，结果如图 3 所示. 从图 3 中可看出：在 DS_1 上，各个模型的预测差异较大，其中｛④、②、①｝预测性能不理想，逐渐下降，其余模型效果较好；在 DS_3 上，8 个模型的预测差异最大，其中偏离中心值最大的依次是｛③、④、⑦、①、②｝，其余的相对较好，但⑧表现得并不优秀，⑥最优秀；在 DS_2 上，各个模型的预测差异整体较小，除较差的｛③、①、②、④｝性能依次降低外，其余 4 个模型预测效果较好；在 DS_4 上，从表现上，可以分为 4 组，其性能依次降低为｛⑥、⑧｝、｛①、②、④｝、｛⑦、⑤｝及｛③｝.

此外，在图 3 中还需指出：1）有些模型的 RE 曲线会出现很高的“突起”，例如，图 3（a）（DS_1）与图 3（c）（DS_3）中. 这主要是由于实际的 TE 数据集中有变动点 CP（change-point）而造成的，例如测试环境的改变就会引发 CP 的出现，使得 CP 前后数据值差距较大，从而引起“突起”现象；2）绝大部分模型在前 2/3 左右的时间内其预测性能一般，之后逐渐趋向良好. 这主要是由于模型中参数较多，只有当数据集增长到一定规模时，才能使模型中的参数逐渐获得接近于最终拟合值的良好效果.

4.4　TEF 性能差异化分析

综合上述实验与分析，从 8 个 TEFs 在 4 组数据集上的度量与预测效果比较的结果可得到如下结论：

1）TEF 对测试数据集的依赖非常强. 由于不同数据集在数据收集方式等上的差异，使得 8 个 TEFs 在有些数据集上表现较好，相反在另外一些数据集上表现一般，差异较大. 例如，在 DS_3 上 8 个 TEFs 的拟合偏差较大；在 DS_1，DS_2 和 DS_4 上各个模型的拟合差异比较小.

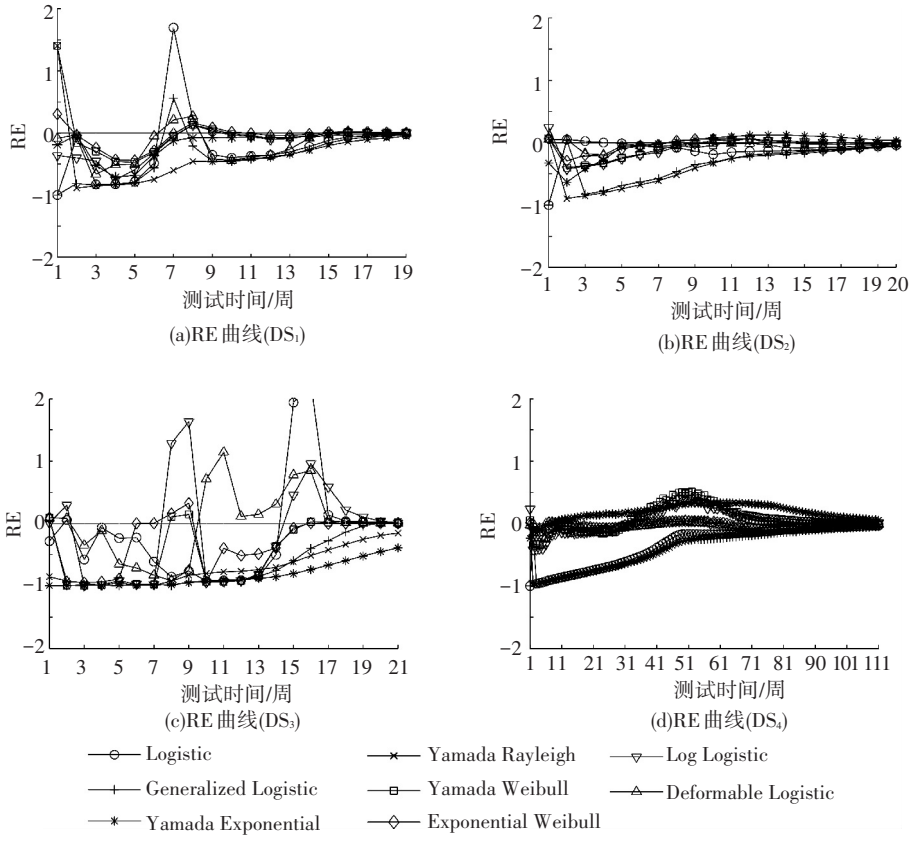

图 3 TEFs 在 4 组数据集上的 RE 曲线比较

2)从拟合与预测的整体效果来看,⑧、⑥、⑤模型的综合性能表现得最优. 其余模型中有的拟合效果较好,但其 RE 预测曲线并不理想;或反之.

3) 基于上述分析,从现有可得的包含 TE 的失效数据集 $DS_1 \sim DS_4$ 可以看出,除⑧、⑥、⑤模型外(仅限在此 4 组数据集上,还需更多数据集实例去验证),并不存在一个最优的 TEF 能够对测试资源的消耗进行精准的管控(度量与预测). 因此,在实际应用中,要应该依据数据集上的差异,合理选择 TEF 进行应用.

4)实际应用中,可以采用"测试数据转换"的方法,这能够从现有数据集上获得较多的测试数据. 例如,一个较为简单的做法是,将现有的以周为单位的测试数据转换为以若干小时为单位,这样数据集的规模会增大,能够缓解"小数据集"带来的不便.

5 结 论

1)TE 表征了测试过程中消耗的资源,与 SRGM 的建模结合促进了 TE 在成本与可靠性工程研究中的重要地位.

2)选择与提出合理的 TEF 可对测试资源进行有效管控,可改善基于 SRGM 所获取的可靠性相关指标特征,从而为建立准确的成本模型与进行最优发布提供决策支持.

3)从对测试资源的描述能力来评价 TEF 过于单一,现有测试的目的已经从传统的排除故障转化到不断提高可靠性为目标,因为软件发布的主要标准之一是其可靠性达到预期. 为此,后续研究中,应对涵盖 TEF 的 SRGM 整体性能进行更为综合的验证与评价.

参考文献

[1] HUANG C Y, LYU M R. Optimal release time for software systems considering cost, testing-effort, and test efficiency[J]. IEEE Transactions on Reliability, 2005, 54(4): 583-591.

[2] JHA P C, GUPTA D, YANG B, et al. Optimal testing resource allocation during module testing considering cost, testing effort and reliability[J]. Computers & Industrial Engineering, 2009, 57(3): 1122-1130.

[3] HUANG C Y, LYU M R. Optimal testing resource allocation, and sensitivity analysis in software development[J]. IEEE Transactions on Reliability, 2005, 54(4): 592-603.

[4] OHTERA H, YAMADA S. Optimal allocation and control problems for software-testing resources[J]. IEEE Transactions on Reliability, 1990, 39(2): 171-176.

[5] HUANG C Y. Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency [J]. Journal of Systems and Software, 2005, 77(2): 139-155.

[6] 张 策, 崔刚, 刘宏伟,等. 软件测试资源与成本管控和最优发布策略[J]. 哈尔滨工业大学学报, 2014, 46(5): 51-58.

[7] BOEHM B W. Software Engineering Economics [M]. New Jersey: Prentice Hall PTR, Upper Saddle River, 1981.

[8] GOKHALE S S, LYU M R, Trivedi K S. Incorporating fault debugging activities into software reliability models: a simulation approach[J]. IEEE Transactions on Reliability, 2006, 55(2): 281-292.

[9] BEMAN O, CUTLER M. Resource allocation during tests for optimally reliable software[J]. Computers & Operations Research, 2004, 31(11): 1847-1865.

[10] HUANG C Y, LO J H. Optimal resource allocation for cost and reliability of modular software systems in the testing phase [J]. Journal of Systems and Software, 2006, 79(5): 653-664.

[11] AHMAD N, KHAN M G M, RAFI L S. A study of testing-effort dependent inflection S-shaped software reliability growth models with imperfect debugging[J]. International Journal of Quality & Reliability Management, 2010, 27(1): 89-110.

[12] KAPUR P K, GOSWAMI D N, BARDHAN A, et al. Flexible software reliability growth model with testing effort dependent learning process [J]. Applied Mathematical Modelling, 2008, 32(7): 1298-1307.

[13] YAMADA S, OHTERA H, NARIHISA H. Software reliability growth models with testing-effort [J]. IEEE Transactions on Reliability, 1986, 35(1): 19-23.

[14] BOKHARI M U, AHMAD N. Software reliability growth modeling for exponentiated Weibull functions with actual software failures data[J]. Advances in computer science and engineering: reports and monographs, 2007, 2: 390-396.

[15] AHMAD N, BOKHARI M U, QUADRI S M K, et al. The exponentiated Weibull software reliability growth model with various testing-efforts and optimal release policy[J]. International Journal of Quality & Reliability Management, 2008, 25(2): 211-235.

[16] PARR F N. An alternative to the Rayleigh curve model for software development effort [J]. IEEE Transactions on Software Engineering, 1980 (3): 291-296.

[17] KUO S Y, HUANG C Y, LYU M R. Framework for modeling software reliability, using various testing-efforts and fault-detection rates [J]. IEEE Trans on Reliability, 2001, 50(3): 310-320.

[18] HUANG C Y, KUO S Y. Analysis of incorporating logistic testing-effort function into software reliability modeling[J]. IEEE Transactions on Reliability, 2002, 51(3): 261-270.

[19] GOKHALE S S, TRIVEDI K S. Log-logistic software reliability growth model [C]//Proceedings of 3rd IEEE International Symposium on High-Assurance Systems Engineering. Washington, DC: IEEE, 1998: 34-41.

[20] ZHANG C, CUI G, MENG F C, et al. A study of optimal release policy for SRGM with imperfect debugging [J]. Journal of Engineering Science and Technology Review, 2013, 6(3): 111-118.

[21] LIN C T, HUANG C Y. Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models [J]. Journal of Systems and Software, 2008, 81(6): 1025-1038.

[22] HUANG C Y. Performance analysis of software reliability growth models with testing-effort and change-point [J]. Journal of Systems and Software, 2005, 76(2): 181-194.

[23] HUANG C Y, KUO S Y, LYU M R. An assessment of testing-effort dependent software reliability growth models [J]. IEEE Transactions on Reliability, 2007, 56(2): 198-211.

[24] AHMAD N, KHAN M G M, QUADRI S M K, et al. Modelling and analysis of software reliability with Burr type X testing-effort and release-time determination[J]. Journal of Modelling in Management, 2009, 4(1): 28-54.

[25] AHMADA N, KHANB M G M, RAFIB L S. Analysis of an inflection S-shaped software reliability model considering log-logistic testing-effort and imperfect debugging[J]. Int. J. Comp. Net. Sec, 2011, 11(1): 161-170.

[26] ZHANG C, CUI G, LIU H W, et al. A unified and flexible framework of imperfect debugging dependent SRGMs with testing-effort [J]. Journal of Multimedia, 2014, 9(2): 310-317.

[27] OHBA M. Software reliability analysis models[J]. IBM Journal of research and Development, 1984, 28(4): 428-443.

[28] WOOD A. Predicting software reliability [J]. IEEE Computer, 1996, 29(11): 69-77.

[29] MUSA J D, IANNINO A, OKUMOTO K. Software reliability: measurement, prediction, application [M]. New York: McGraw Hill, 1987.

[30] PHAM H. Software Reliability [M]. Singapore: Springer, 2000.

（编辑　张　红）