ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.12, December 2010, pp.2999−3010
doi: 10.3724/SP.J.1001.2010.03691

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

软件缺陷发现时序过程的叠加双阻尼振荡模型*

何智涛+, 晏海华, 刘 超

(北京航空航天大学 计算机学院,北京 100191)

Accumulative Bi-Damped Oscillation Model for the Sequential Process of Software Defect Discovery

HE Zhi-Tao+, YAN Hai-Hua, LIU Chao

(School of Computer Science and Engineering, BeiHang University, Beijing 100191, China)

+ Corresponding author: E-mail: zhitaohe@vip.sina.com

He ZT, Yan HH, Liu C. Accumulative bi-damped oscillation model for the sequential process of software defect discovery. *Journal of Software*, 2010,21(12):2999−3010. http://www.jos.org.cn/1000-9825/3691.htm

Abstract: In software testing practice, usually, the software under test (SUT) experiences multiple iterative processes of testing and modification. With the influence of many uncertain factors, such as defect distribution in the software under test, iterative developing and testing processes, and testers' capabilities of defect detection, the software defect discovery process shows that the sequential characteristics correspond with test cycles, i.e. periodicity, random oscillation, and attenuation. Through a deep analysis of the basic characters and key influencing factors of the controlled software testing process, the paper proposes an Accumulative Bi-Damped Oscillation Model (ABDOM), which describes periodicity, random oscillation, and attenuation for a sequential software defect discovery process. It verifies ABDOM's validity with defect data gathered from 2 true software test projects, and discusses the application scope of ABDOM and the possible applications in test predication and evaluation.
Key words: software defect; defect detecting; defect discovery model; sequential

摘 要: 在软件测试实践中,被测软件通常要经历多轮次的测试和修改过程.由于受到被测软件的缺陷分布、迭代式的开发与测试过程、测试者发现缺陷的能力等诸多非确定性因素的影响,使得软件缺陷发现的时序过程呈现出相应的周期性、随机振荡性和阻尼衰减等时序特征.通过对以软件缺陷发现为目标、测试过程管理规范的软件测试过程基本特征和关键影响因素的深入分析,提出了一种描述软件缺陷发现时序过程特征的叠加双阻尼振荡模型(accumulative bi-damped oscillation model,简称 ABDOM).采用从两个真实软件测试项目中采集的缺陷发现过程数据,检验了 ABDOM 模型的有效性,定义了 ABDOM 模型的适用范围,并对 ABDOM 模型的应用进行了初步讨论.
关键词: 软件缺陷;缺陷发现;缺陷发现模型;时序
中图法分类号: TP311 **文献标识码**: A

长期以来,困扰软件测试者的一个难题是,如何对软件测试的进展状况进行评估.对于一个复杂的软件测试过程,测试者需要及时地掌握其进展状况,以便随时调整测试重点、测试策略和测试资源,避免盲目的测试,提高

* Supported by the National Natural Science Foundation of China under Grant No.90718018 (国家自然科学基金)
Received 2009-02-11; Revised 2009-05-21; Accepted 2009-07-09

测试效率和控制测试成本,同时也有助于判断测试是否能够如期完成和评估产品的质量.

在软件测试实践中,通常可以从两个方面来判断软件测试工作的进展状况:一是分析测试计划的完成情况,其主要依据是测试的覆盖程度,或称作测试的充分性,也就是依据覆盖准则来确定测试内容,并在测试过程中做记录,据此,随时可以了解到测试的进度和剩余工作.一般而言,确保测试的充分性是必要的,因此为确保充分测试所需的工作量估计成为制定测试计划和评估测试进展的重要依据.但是,测试的充分性是相对的.在有限的资源和时间约束下,测试充分性只是针对被测软件某些特征和性质的有限覆盖,如语句覆盖、功能点覆盖、运行环境和应用场景的覆盖等.因此,软件测试需要在预定的测试计划或测试方案指导下,根据测试过程中发现的新问题以及被测软件演化情况等,适时进行动态调整.因此,单纯依据预先制定的测试计划和测试大纲来评估测试的进展显然是不充分的;二是观察软件缺陷的发现率,或者说是通过估计残留缺陷来评估软件测试进展.测试目的在于发现软件中暗藏的缺陷[1].尽管我们难以准确地获知软件中的残留缺陷,但是通过对测试过程的追踪和分析,特别是通过分析软件缺陷发现的数量、严重性和分布情况等,可以揭示软件缺陷发现过程的一般规律,并据此来估计残留缺陷的情况,从而判断软件测试的进展状况和评估被测软件的质量.

需要说明的是,由于被测软件自身的复杂性和演化性,以及测试资源(如人力等)和时间的有限性,软件测试不仅要关注覆盖率,还需重视测试效率.即在兼顾测试覆盖面的同时也要突出测试重点,以期提高发现缺陷的可能性,并控制测试资源和时间等方面的代价.因此,在受限资源和规定时限约束下提高发现新缺陷的可能性,成为指导软件测试的重要原则.然而,由于被测软件的缺陷分布、迭代式的开发与测试过程、测试者发现缺陷的能力等诸多非确定性因素的影响,基于测试的软件缺陷发现过程呈现出随机振荡的时序特性,因此难以简单利用缺陷发现数量的时序变化曲线来评估软件测试的进展情况和预测后续测试发现新缺陷的可能性.事实上,目前尚无一种简明方法来帮助测试者估计软件测试的进程和预测软件的质量.

我们注意到,在软件开发过程中,被测软件总是要经历多轮次的测试和修改过程.同时,从 10 多年的软件测试工程实践中也发现,在一个以发现软件缺陷为目标的、测试过程管理较为规范的测试过程中,被测软件的缺陷分布、迭代式的软件开发和测试过程以及与测试者的经验和投入相关联的缺陷发现能力等非确定性因素对软件缺陷的发现过程有着显著影响,使其表现出与测试轮次相关联的周期性、因各种复杂因素导致的随机振荡性以及随着越来越多的缺陷被发现和修复而表现出来的缺陷发现阻尼衰减性等时序特征.本文通过对软件测试过程的深入剖析,分析了软件缺陷发现时序过程的周期性、随机振荡性和阻尼衰减性等基本特征及其关键影响因素,提出了一种可比较准确描述软件缺陷发现时序过程的数学模型,称作叠加双阻尼振荡模型(accumulative bi-damped oscillation model,简称 ABDOM),并用两个真实软件测试项目中采集的缺陷发现数据对该 ABDOM 模型的有效性进行了检验.

本文第 1 节简要介绍有关软件缺陷发现模型的研究现状.第 2.1 节重点分析一般软件测试过程的基本特性;第 2.2 节深入分析在软件测试过程中软件缺陷发现过程的时序特性及其关键影响因素,并据此提出一种能够描述软件缺陷发现时序过程的数学模型 ABDOM;第 2.3 节对 ABDOM 模型的适用范围进行说明.第 3 节简要分析采用两个软件测试项目的缺陷发现过程数据对该模型进行检验的结果.第 4 节是 ABDOM 模型应用讨论.第 5 节总结全文.

1 有关软件缺陷发现模型的研究现状

多年来,针对软件缺陷模型的研究很多,但是这些研究主要集中在对软件缺陷总量的预测方面.这些模型大致分为以下几类:

(1) 根据程序源代码的规模、复杂性进行缺陷估计的方法[2−4],如Halstead缺陷预测模型[5];

(2) 根据对以往同类软件中所含缺陷数据的分析来预测当前被测软件中缺陷数量的方法,发表的模型有以下几种:由Ganhey和Davis提出的基于阶段的模型Phase Based Model[6]、Marryland大学的Carol等人提出的一种早期预测的方法[7];

(3) 根据前期测试获取的已知缺陷或者预先植入的缺陷被发现的比例来预测软件中剩余缺陷数量的方

法,如捕获-重捕获方法的预测模型[8,9];

(4) 根据神经网络预测理论,如基于神经网络的预测模型[10,11]、基于Bayes网的软件缺陷预测模型[12−14].

这些模型可以辅助测试者对测试的充分性和可信性进行一定意义上的评价,但是这些模型都是基于某些假设前提对被测产品缺陷的总量进行预测或计算.比如:Halstead 方法是根据代码中的操作符和操作数的类型和数量来估计缺陷的数量,这是基于软件的(操作指令)复杂度越高存在缺陷的可能性也越高的假设;根据以往项目的缺陷数据来进行预测的方法则认为,同一个软件组织开发同样的软件,其缺陷分布大体相同;捕获-重捕获方法是根据软件中的已知缺陷在测试中被发现的比率来估计软件中缺陷的总量,其认为,这些缺陷(不论其是否是已知的)在测试中被发现的概率相同;基于神经网络和 Bayes 网的预测模型则是利用程序中各种成分或性质之间存在的逻辑关联来进行推理和演算.可见,这些模型无法反映因软件的类型差异和个体差异、软件开发与测试方法的差异及其他诸多复杂因素对被测软件中残留缺陷分布的综合影响,故难以被广泛应用.更为重要的是,面对复杂的迭代式软件开发与测试过程,这些模型无法描述由于被测软件的不断修改和扩展所带来的缺陷变化,比如已有缺陷的修复和新缺陷的引入.为此,有必要深入研究软件缺陷发现过程的时序特性,构建一种能够准确刻画软件缺陷发现过程时序特性的数学模型,揭示软件的缺陷分布、迭代式的软件开发与测试、软件测试者的能力等非确定性因素对软件缺陷发现时序过程的影响规律,以帮助测试者对软件测试的进程进行量化的评估和预测.

2 基于测试的软件缺陷发现过程及其特性

在本文中,软件缺陷泛指被测软件中存在的各种问题,包括在功能、可靠性、性能、易用性、安全性等方面存在的或者表现出来的问题.软件测试过程是指在软件开发过程中与测试直接相关的一系列活动,比如单元级测试、集成测试、独立第三方测试等.软件缺陷发现的时序过程(sequencial process of software defect discovery,简称 SPSDD)特指在软件测试过程中单位时间(如每日)内发现缺陷数量的时序关系.

人们期望这种软件缺陷发现的时序过程是一个平稳的过程,可以据此来比较准确地掌控测试的进程,评估和预测被测软件的质量状态.如果每天的缺陷发现数量的上限在经历一个简单的增长过程之后便转为单调递减且在预期的时间 t 内稳定趋近于 0,如图 1 所示.不难判定,就图 1 中所示的测试而言,在预期的时间 t 之内没有发现更多的软件缺陷,在此软件缺陷分布基础上,如果此测试项目中的测试需求分析和测试用例覆盖被证明是充分的,则不难评估测试过程的质量状况.

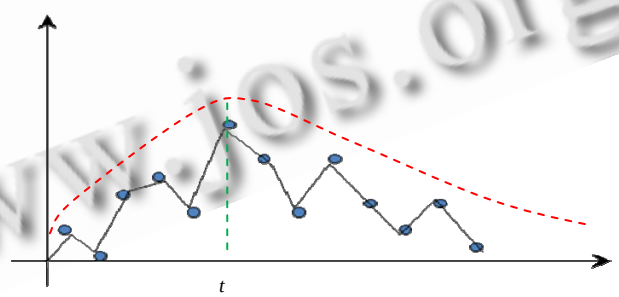

Fig.1 Expectation curve of sequential process of software defects discovery

图 1 软件缺陷发现时序过程的期望(上限)曲线

然而在软件测试实践中,软件缺陷发现过程呈现出某种周期性随机振荡的时序特性.图 2 是一个真实的软件系统在测试期间的软件缺陷发现过程,显然,仅凭(图中虚线所示的)不规则的周期性振荡曲线很难对测试进程和软件缺陷发现的变化趋势作出推断.因此,构建能够准确刻画这种时序特性的数学模型将有助于发现影响其时序特征的重要因素,从而揭示其内在规律.

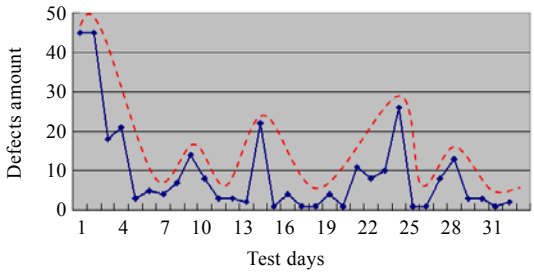

Fig.2 Project *A*'s sequential distribution graph of software defects discovery

图 2 某公文系统 *A* 的软件缺陷发现时序分布图

2.1 软件测试过程及其特性

在软件开发实践中,软件测试是一个复杂的、受诸多非确定性因素影响的过程,表现出如下重要特性:

(1) 时序性:无论采用什么样的测试方法,软件测试都要经历很长的时间,比如数周、数月或更久.一个规范的软件测试过程包含测试需求的获取与分析、测试计划的制定、测试大纲及测试用例的设计与生成、测试的实施、测试结果的分析和软件问题(或缺陷)的报告,以及回归测试等一系列活动[15].这些活动之间既存在一定的时序约束,又可以相对独立、彼此并行地执行.因此,软件测试具有复杂的时序特性.

(2) 并发性:无论采用哪种软件开发范型,如瀑布模型、迭代式开发过程等,软件测试实际上是一个与软件需求分析、软件设计、编码实现等开发活动并发执行的过程,特别是与缺陷修复、功能变更等活动之间的交替或并行关系,使得被测对象具有动态演化的特性.

(3) 周期性:根据软件的规模和质量要求,测试过程可以被划分为单元测试、配置项测试、集成测试等多个测试阶段,也可以被分解为功能测试、性能测试、接口测试等多种类型.特别是伴随着缺陷的修改以及新功能的实现或者扩展,软件测试需经历多个轮次,故具有周期性特征.

(4) 非确定性:鉴于软件缺陷的分布、迭代式的软件开发与测试、软件测试者的能力等非确定性因素的影响,使得测试过程具有"事件驱动"的性质,故表现出局部的"随机性"或动态性特征.

(5) 时限性:软件测试的充分性受到有限资源和规定时限的制约,随着测试的进行,必须适时地调整测试策略,及早发现和修复各种缺陷,保证软件测试按期完成.因此,软件测试的充分性是相对的,测试具有渐进性特征.

软件测试过程的上述特性对软件缺陷发现过程的时序性质具有直接影响,使其具有如下特性:

(1) 时序性:通过测试发现软件缺陷的数量具有非确定性的时序分布规律.

(2) 周期性:软件缺陷发现数量曲线有多个周期性峰值和接近于 0 的低谷,这种周期性与测试周期有关;

(3) 衰减性:随着测试的持续,特别是随着已发现缺陷的修复,新发现的软件缺陷数量最终会呈现衰减趋势,并逐渐趋于 0.

(4) 振荡性:受各种非确定性因素影响,该曲线表现出随机振荡的特点,从而干扰了其周期性和衰减性.

这种不规则的周期性和振荡性源于诸多非确定性因素的综合影响,包括:

(1) 被测软件的影响因素:被测软件的规模、复杂性及其成熟度预示了暗藏缺陷的数量、类型和分布,反映了软件缺陷发现的难易程度.

(2) 开发过程的影响因素:软件开发过程的迭代性导致被测软件自身的演化,使得被测软件(版本)及其所隐藏的缺陷都是变化的.即在软件演化过程中,有些缺陷被修复,同时一些新缺陷有可能被注入;

(3) 软件测试的影响因素:软件测试资源和时间的约束、软件测试者(或测试部门)的测试能力、测试方法和测试策略(如测试任务的执行顺序)等.

根据多年来的软件测试实践的跟踪和分析,我们发现这些非确定性因素可被归结为若干个关键影响因子.

这些因子具有明确的直观含义,测试者可用简单的方法给出一个初始的(近似)估计值.更重要的是,在测试过程中需要根据软件缺陷发现过程的实际数据,通过与缺陷发现时序过程模型的拟合对这些影响因子的取值加以修正,以便对该过程的当前进展状况和后续变化趋势进行更加准确的评估和预测.

2.2 软件缺陷发现的时序过程及其关键影响因素

2.2.1 基于被测软件版本序列的软件缺陷发现时序过程模型

如上所述,被测软件为 S 包含着一个版本演化的序列,记作

$$S=\{V_i,i=0,\ldots,n\} \tag{1}$$

其中,令V_0为空,记作$V_0=\theta$,表示软件S起始点的空版本;令V_n为经过测试后的正式提交版.各版本的提交时间t_i构成的序列称作该软件的版本演化周期,记作

$$T(S)=\{t_i,i=0,\ldots,n\} \tag{2}$$

其中,令$t_0=0$,作为在第 1 个测试版本V_1提交之前的起始时间,记作$t_0=0$;t_n为测试版本正式提交时间.

将基于测试的软件缺陷发现时序过程记作 $D(S,t)$,表示在时间 t 通过测试发现的缺陷数量.根据公式(1)可得

$$D(S,t)=\{D_i(V_i,t|t\in[t_i,t_{i+1}]),i=0,\ldots,n-1\} \tag{3}$$

其中,不妨令$D_0(V_0,t)=0$,表示第 1 个测试版本V_1提交之前没有软件缺陷报告.

由于每当提交一个新版本V_i时,它都将替代前一个版本V_{i-1},因此,针对V_{i-1}的测试随即终止,后续测试转而针对V_i.从这个角度看,$D(S,t)$可以根据被测软件的版本演化周期划分为若干个彼此独立的、分时段的子过程,因此有如下关系式成立:

$$D(S,t)=D_i(V_i,t) \text{ iff } t\in[t_{i-1},t_i],\ i=1,\ldots,n-1 \tag{4}$$

公式(4)被称作基于被测软件版本序列的软件缺陷发现时序过程模型(V-SPSDD).

在测试实践中,测试者可以准确地记录和跟踪被测软件S的各个版本在对应的测试周期中发现的缺陷数量.因此,采用这种分时段的方法来描述软件缺陷发现时序过程可以建立软件缺陷发现与软件版本之间的直接对应关系,报告每个版本(在对应的测试周期内)的缺陷发现情况.但是,这种方法割裂了各个版本之间以及对应的各组测试之间的内在关联.比如:V_i继承了V_{i-1}的大部分功能及各种特性,因此对V_i的测试不必重复在前面已经做过的大部分测试,除非某些的部分被修改或可能受到某些修改的影响;对V_i的某项测试也可看作是对其老版本对应功能或特性的测试,除非被测项是在新版本中增加的或者被修改过.因此,采用这种方法难以根据在整个测试过程中发现软件缺陷的综合情况对测试的进展和后续进程进行合理的评估和预测.

2.2.2 基于被测软件版本变更序列的软件缺陷发现时序过程模型

在分析软件测试的进展时,我们注意到新版本的变化量是影响后续测试进程的关键因素.

软件的变更具有广泛的涵义.一般而言,软件的变更域包括增、删或修改的程序单元、具体的功能项或者某种特性.对于被测软件S,其新版本V_i相对于前一个版本V_{i-1}的变更部分称作变更域;将因受变更影响可能引发新问题而必须重新测试的部分称作波及域;将V_i中的变更域和波及域的并集称作V_i中的关键(测试)域,记作Δ_i;将非关键域部分称作这两个相邻版本V_i和V_{i-1}的稳定域,记作B_i.为此,关于被测软件的版本变更序列$\{\Delta_i\}$,可以给出下列关系式

$$\left.\begin{aligned}
&\Delta_i=V_i\square V_{i-1}=(V_i-B_i)=\{v_{ij}\mid v_{ij}\text{表示}V_i\text{关键域中的程序单元、功能项或者具体特征},j=1,...,m_i\}\\
&B_i=B_{i-1}=V_{i-1}-V_{i-1}\cap\Delta_i\\
&V_i=V_{i-1}\oplus\Delta_i=B_i\cup\Delta_i=\bigcup_{k=1}^{i}\Delta_k,i=1,...,n
\end{aligned}\right\} \tag{5}$$

其中,$\square$和$\oplus$是两个特殊的关系运算符,分别表示两个实体的差异部分和对差异部分的替换.将V_i分解为B_i和Δ_I两部分,有助于将V_i的缺陷发现时序过程分解为两个对应的部分.所以,被测软件S的软件缺陷发现时序过程可以表示为

$$D(S,t)=\sum_{i=1}^{n}D(\Delta_i,t)=\sum_{i=1}^{n}D(\Delta_i,t\overset{\oplus}{-}t_i) \tag{6}$$

其中,算子$\overset{\oplus}{-}$表示当$t\geqslant t_i$时,$t\overset{\oplus}{-}t_i=t-t_i$,否则等于 0.$D(\Delta_i,t\overset{\oplus}{-}t_i)$表示自周期$t_i$开始测试所发现的与$\Delta_i$有关的缺陷数量的时序关系.公式(6)称作基于被测软件版本变更序列的软件缺陷发现时序过程模型(D-SPSDD).在公式(6)中,将Δ_i视作自t_i之后始终存在,即忽略了在t_{i+1}之后$\{\Delta_k,k=1,\ldots,i\}$中被$\Delta_{i+1}$替代部分不再会被测试到,并因此不再会在其中发现新缺陷的情况.这种缺陷属于在前期测试中未发现但在新版本中已"被间接修复"的缺陷.显然,这是对被测软件测试域的少许放大,因为通常情况下,这种情况所占比重很小,可以忽略不计,不会影响该模型对测试进展的估计.

2.2.3 基于被测软件版本关键域的软件缺陷发现时序过程模型

由于Δ_i是相对稳定不变的,不受后续版本变更的影响.所以,对Δ_i的测试将不会受到与版本变更相对应的测试周期的影响.针对Δ_i的软件缺陷发现时序过程主要受到Δ_i本身的规模和复杂度、测试者发现缺陷的能力、缺陷发现时间的随机性等因素的影响,同时,随着缺陷逐步被发现,其中的残留缺陷数量会减少,致使发现新缺陷的难度增加,时间延长.因此,Δ_i的软件缺陷发现时序过程为

$$D(\Delta_i,t_i,t)=R_iA_ie(a_i,t_i,t)g(\omega(t,t_i),t,\varphi_0),t\geqslant t_i \tag{7}$$

其中,

(1) R_i是Δ_i的规模及其复杂度因子,比如

$$R_i=C_i|\Delta_i|/|V_i| \tag{8}$$

或者,更精细地采用 $R_i=\sum C_{ij}(|v_{ij}|/|V_i|)$.显然,软件变更规模大、复杂度越高,则隐藏的缺陷越多.

(2) A_i是测试者发现缺陷的能力因子,是对测试者个人的缺陷发现能力、测试组织的管理能力等的综合评估.一种简捷、直观的表示方式是测试者每日能够发现的软件缺陷数量的最大值,记作$A_{\max}$.显然,测试者的能力越强,测试前期发现缺陷的数量越多.但是,考虑到在测试过程中,由于测试方法、手段、策略以及测试区域等的调整对缺陷发现能力的影响,A_i也可以是一个时序函数.

(3) $e(a_i,t_i,t)$是缺陷发现数量的衰减因子,a_i表示发现新缺陷数量上限的衰减速率.随着越来越多的缺陷被发现,残留缺陷逐渐减少,因此,新发现的缺陷数量也随之下降,并逐步趋于 0.对于一个具有稳定测试能力的规范的软件测试过程,如果其发现新缺陷的数量上限在统计意义上具有单调下降的趋势且最终趋于 0,则不妨选择

$$e(a_i,t_i,t)=\begin{cases}\mathrm{e}^{-a_i(t-t_i)}, & t\geqslant t_i\\ 0, & t<t_i\end{cases} \tag{9}$$

其中,令$a_i=a$,即对各测试周期而言,测试的稳定性是一致的.

(4) $g()$是一个随机振动函数.当按计划依次对各个功能域或特征域进行测试时,这种振动具有相对稳定的周期τ,故可选用sin()作为振动函数,且有确定的初始相位$\varphi_i=\varphi$和初始频率 $2\pi/\tau$.

$$g(t,t_i)=\sin(\omega(t,t_i)+\varphi)=\sin\left(\frac{2\pi}{\tau}(t-t_i)+\varphi\right).$$

一般地,若考虑测试中各种非确定性因素的影响,则有

$$\omega(t,t_i)=\frac{2\pi}{\tau}f(b_i,t_i,t).$$

如果在一定的时间(如t_i的c倍时间)之后,振动频率将逐步衰减,则可以选择一个衰减函数,比如

$$f(b_i,t_i,t)=\begin{cases}\mathrm{e}^{-b(t-ct_i)}, & t\geqslant t_i,b_i=b\\ 0, & t<t_i\end{cases} \tag{10}$$

故有

$$g\left(\frac{2\pi}{\tau}f(b,t_i,t)t+\varphi_0\right)=\begin{cases}\sin\left(\frac{2\pi}{\tau}\mathrm{e}^{-b(t-ct_i)}(t-t_i)+\varphi_0\right), & t\geqslant t_i\\ \sin(\varphi_0), & t<t_i\end{cases} \tag{11}$$

测试实践中,如果采用规范的测试方法,具有稳定的测试过程,则 $f()$应当是一个有规律的,其频率是稳定的(或稳定变化的);否则,说明有过多非确定因素的严重干扰,测试过程需要改进.

由于$\Delta_1=V_1$,所以不难看出,公式(7)实际上是针对单一版本的被测软件进行测试产生的软件缺陷发现时序过程的模型.

2.2.4 面向被测软件版本变更序列的软件缺陷发现时序过程模型

根据上述分析,不难得出面向被测软件版本变更序列$\{\Delta_i\}$的软件缺陷发现时序过程模型:

$$D(S,t)=\sum_{i=1}^{k}D(\Delta_i,t\overset{\oplus}{-}t_i) \tag{12}$$

其中,当$t\geqslant t_i$时$t\overset{\oplus}{-}t_i=t-t_i$,否则$t\overset{\oplus}{-}t_i=0$;累加次数$k(1\leqslant k\leqslant n)$表示当前的版本序号.不妨将$n$定义为计划或实际的软件测试轮次总数,而 k 定义为当前的测试轮次.尽管在一轮测试中可能会多次更新被测版本,但这种更新通常只涉及特定缺陷的修复或特定功能的扩展.因此,将其视作同一个版本不会对缺陷发现时序过程的估计造成严重的偏差.此外,鉴于本文并不关注两个相邻的测试轮次之间可能存在的时间间隔(如果存在可以滤掉),故本文约定测试周期之间是连续的.

对于一个有多年测试经验的软件测试项目组,在承接软件测试项目时将遵循一定的软件测试过程规范,在对被测软件充分熟悉的前提下,制定完整和详细的测试计划,设计相对充分的测试用例集合,并且按计划对被测软件进行连续或相对集中的多轮次测试.尽管测试过程会受到软件需求变更、版本提交时间变更、测试策略和测试重点调整,甚至人员流动等复杂因素的影响,但是可以预期,在一个以软件缺陷发现为目标、测试过程管理规范的软件测试中,软件缺陷发现时序过程具有以下特点:

(1) 测试项目组的能力相对稳定,故不妨用单日可发现缺陷最大数$A_{\max}$表示.例如,一个由 4 名测试工程师组成的测试组,如果平均每人每天最多可发现 25 个缺陷(包括填写缺陷报告等相关工作),则$A_{\max}=100$.

(2) 具有明确的测试周期$T=\{t_i,i=1,\ldots,n\}$.

(3) 对于开发部门提交的每一个新版本V_i,可以给出R_i的一个估计值($|\Delta_i|/|V_i|$).一般情况下,不妨令公式(8)中的$C_i=1$.

(4) 在各轮次的测试中,软件缺陷发现曲线的衰减率是大体一致的,均为e^{-at},其中的阻尼系数a反映了测试需持续的时间长度,是评估和选择合理的测试持续时间的重要参数.显然,a的值越大,说明缺陷发现的速度越快.例如当$a=0.3$时,如果测试的基本时间单位是天,则采用同样的测试方法持续测试 10 天与测试 20 天甚至 30 天所发现的缺陷数量之比约在 95%以上.参数a的取值受到测试者发现缺陷的速度和被测软件变更量的影响.

(5) 在各轮次的测试中,软件缺陷发现曲线的振动频率大体一致,且均呈下降趋势并逐渐趋于平稳,衰减率为e^{-bt}.参数b的取值反映了测试组发现缺陷过程的稳定性.当b很小时,可近似为 0,表明缺陷发现过程具有稳定的周期性,比如平均每 3 天左右(即$\tau=3$)会出现一次发现缺陷的高峰.

根据以上分析及其公式(7)和公式(12)可得,基于测试的软件缺陷发现时序过程模型如下:

$$D(S,t)=\sum_{i=1}^{k}D(\Delta_i,t\overset{\oplus}{-}t_i)=A_{\max}\sum_{i=1}^{n}R_i e^{-a(t\overset{\oplus}{-}t_i)}\sin\left(\frac{2\pi}{\tau}e^{-b(t\overset{\oplus}{-}t_i)}(t\overset{\oplus}{-}t_i)+\varphi_0\right) \tag{13}$$

该模型被称作叠加双阻尼振荡模型.

第 2.1 节给出的软件缺陷发现时序过程的特性(时序性、周期性、衰减性和振荡性)在叠加双阻尼振荡模型ABDOM中均得到了较好的描述:软件缺陷发现的时序性和周期性通过$\sin(t)$函数表达;软件缺陷发现的衰减特性通过指数函数e^t表达;振荡特性则通过双阻尼模型的叠加进行描述.

产生软件缺陷发现振荡特性的影响因素(被测软件影响因素、开发过程影响因素和测试过程影响因素)也在ABDOM模型中得到反映:$A_{\max}$,b和τ反映了测试过程影响因素,既$A_{\max}$描述测试项目组的能力,也从侧面反映了软件测试资源状况,τ描述软件缺陷发现周期,b用于描述软件缺陷发现周期的稳定性;R_i描述开发过程的影响

因素,即新版本引入代码规模;a描述软件缺陷发现数量衰减率,揭示了被测软件中发现软件缺陷的难易程度和被测软件代码质量的影响,也在一定程度反映了软件测试的质量.

2.3 ABDOM模型适用范围说明

ABDOM 模型可以较好地反映软件缺陷发现时序过程的主要特性,即时序性、周期性、衰减性和振荡性.当从一个测试过程中无法获取测试组织缺陷发现能力、缺陷发现周期、软件缺陷发现衰减率、软件版本变更等信息时,ABDOM 模型将无法适用于该测试过程.因此,ABDOM 模型适用于表达测试资源配置稳定的、测试过程和开发过程控制严格的、以发现软件缺陷为目标的软件缺陷发现过程;而当测试组织松散,缺少测试过程管理时, ABDOM 模型将无法很好适用.如开源社区中的测试活动、组织松散且缺少周密的测试计划与过程管理、没有稳定的测试资源配置,无法用 ABDOM 模型来准确刻画其特性;以测为辅、以评为主的各种不以软件缺陷发现为主要目标的软件测评活动、项目验收测试等也不适用 ABDOM 模型描述.

3 ABDOM 模型的检验

北京航空航天大学软件工程研究所研发的软件测试管理平台 QESuite 可支持测试组织对测试项目及其测试过程进行有效的管理,特别是通过详细记录各项测试活动产生的测试需求、测试计划、测试用例、软件问题(缺陷)报告、测试执行结果报告等各种测试制品及其状态信息实现对整个测试过程的实施追踪与监控.近年来, QESuite 产品已在一些科研机构和软件企业以及一批信息管理系统测试中得到成功应用.本节提供两个应用 QESuite 的真实测试项目缺陷发现时序过程的数据,对公式(13)与实践数据的拟合效果进行检验,并对结果进行分析.这两个真实测试项目中的测试活动均符合 ABDOM 模型的适用标准,即均为以发现软件缺陷为目标的、组织稳定、测试过程管理较为规范的测试活动.

本节对模型的验证采用了两种方式:一是使用一个在迭代式测试项目中收集的缺陷发现过程的时序信息来验证模型对增量式、非单调衰减的软件缺陷发现过程的拟合情况,检验模型的合理性;二是利用一个非增量式测试项目中收集的前期软件缺陷发现数据,使用模型拟合前期数据,然后对其后期测试进行预测,并通过与后期软件缺陷发现真实数据的比对.

3.1 增量式软件开发的测试过程拟合

某公文系统 A 是某机关的核心系统,由于涉及数据的敏感性,要求软件系统投产后不再维护,对软件质量要求非常高.其第三方测试过程是一个迭代测试过程,被测系统的大部分模块在测试前已经开发完成,测试方投入经验丰富的测试团队对其进行全面测试.开发方在测试过程中陆续集成新功能模块,导致测试方必须进行多次的补充测试和回归测试,使软件缺陷发现呈现频繁振荡的特点.软件缺陷发现情况见表 1,软件缺陷时序发现分布图如图 2 所示.

Table 1 Project *A*'s software defects amount by found date

表 1 某公文系统 *A* 软件缺陷按时序发现数量统计

Found date	Defects amount	Found date	Defects amount	Found date	Defects amount
2005-04-26	45	2005-05-27	3	2005-11-22	10
2005-04-27	45	2005-05-30	2	2005-11-23	26
2005-04-28	18	2005-06-06	22	2005-11-24	1
2005-04-29	21	2005-06-07	1	2005-11-27	1
2005-04-30	3	2005-06-09	4	2005-11-28	8
2005-05-10	5	2005-06-12	1	2005-11-30	13
2005-05-11	4	2005-06-13	1	2005-12-01	3
2005-05-17	7	2005-06-14	4	2005-12-02	3
2005-05-18	14	2005-06-16	1	2005-12-03	1
2005-05-19	8	2005-06-22	11	2005-12-04	2
2005-05-26	3	2005-11-21	8	2005-12-05	0

采用均方差最小值拟合法,与公式(13)自动拟合,获得了较好的拟合结果,如图 3 所示.

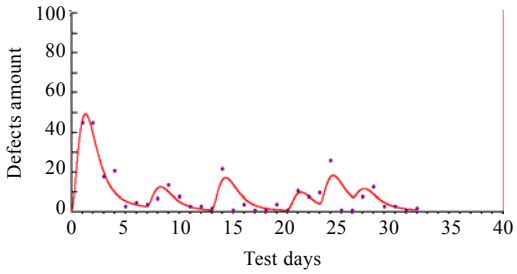

Fig.3 Automatic fitting result of project *A*'s defects discovery data by ABDOM

图 3 ABDOM 对某公文系统 *A* 软件缺陷发现数据的自动拟合结果

在进行自动拟合过程中,选取的各参数取值范围如下:

- t: t=0,1,2,3,…,32.
- A_{max}: 30<A_{max}<120,A为整数.
- τ: 2<τ<7,τ为整数.
- a: 0<a<1.
- b: 0<b<1.
- R_1~R_6: 0<R_i≤1.
- t_1=0,t_2=7,t_3=13,t_4,t_5=23,t_6=26.

通过自动拟合获得的各参数值及对其直观意义的分析如下:

(1) 缺陷发现速率A_{max}=99,表示测试组每日发现缺陷数量的上限,基本符合其软件缺陷发现能力(该测试组在过去 2 年里承担的多个测试项目中所表现出的缺陷发现能力在 80 个/日~110 个/日左右);

(2) 缺陷发现周期系数τ=2.6,基本符合该测试团队缺陷发现周期为 3 日的实际情况;

(3) 对应于 6 个测试版本与测试轮次,代码变动比例基本符合实际情况:

- R_1=1.00,第 1 个版本;
- R_2=0.22,部分功能作调整;
- R_3=0.34,新增指纹识别功能;
- R_4=0.19,界面改版;
- R_5=0.34,更新指纹识别模块;
- R_6=0.17,个别功能作调整.

(4) 缺陷发现阻尼 a=0.45,阻尼系数较高,表明软件测试的质量较高,缺陷收敛度较高.2009 年,对该公文系统应用情况回访证实,公文系统 *A* 在投产 3 年中持续稳定运行,未出现故障,用户非常满意;

(5) 缺陷发现周期阻尼 b=0.197,阻尼为正,表明出现软件缺陷高峰的周期在不断延长.

公式(13)与该项目实测数据的拟合效果很好,最小均方差为 *Error*=5.04.对于该项目而言,公式(13)的拟合结果是令人满意的,各参数的取值也基本符合测试过程中所记录的相关特征信息.

3.2 ABDOM的预测能力验证

ABDOM 具备一定的缺陷发现预测能力.测试过程中,在将测试前期获得的软件缺陷发现时序数据输入 ABDOM 模型进行拟合的基础上,可对测试后期的软件缺陷发现分布状况进行预测.

某政务信息系统 *B* 的测试过程中进行了一轮测试(共 11 天)和一轮回归测试(6 天).事隔 1 年后,又进行了测试(6 天).其软件缺陷发现数量时序关系见表 2.

Table 2 Project *B*'s software defects amount by found date

表 2 某政务信息系统 *B* 软件缺陷按时序发现统计

Found date	Defects amount	Found date	Defects amount	Found date	Defects amount
2004-10-14	20	2004-10-26	12	2004-11-25	3
2004-10-15	16	2004-10-28	7	2005-07-28	1
2004-10-18	13	2004-10-29	1	2005-07-29	1
2004-10-19	16	2004-11-09	2	2005-08-16	4
2004-10-20	19	2004-11-10	7	2005-08-17	12
2004-10-21	15	2004-11-11	11	2005-08-18	2
2004-10-22	5	2004-11-23	7	2005-08-20	1
2004-10-25	12	2004-11-24	6		

为了验证 ABDOM 的预测有效性,这里首先对测试前 17 天(第 1 轮测试+回归测试)的软件缺陷发现数据进行自动拟合,所得曲线如图 4 所示,所得各参数为

缺陷发现速率A_{max}=28,,缺陷发现阻尼a=0.07,缺陷发现周期系数τ=2.84,缺陷发现周期阻尼b=0.026.

公式(13)与某政务信息系统 *B* 前 17 天的缺陷发现数据的拟合效果很好,最小均方差为 *Error*=3.13.

其中,缺陷发现速率A_{max}为 28,基本符合该测试团队在 2004 年的能力峰值;缺陷发现周期系数为 2.86,基本符合该测试团队的缺陷发现周期特点;缺陷发现阻尼为 0.07(相比公文系统*A*的缺陷发现阻尼为 0.45),暗示某政务信息系统*B*的测试工作的质量不如某公文系统*A*的测试质量;缺陷发现周期阻尼b=0.026(相比公文系统*A*的缺陷发现周期阻尼为 0.197).缺陷发现阻尼和缺陷发现周期阻尼同时较小,说明该软件系统中仍然可能有较多的软件缺陷剩余.

拟合结果中 17 天之后的曲线揭示,在后续测试中仍能发现较多数量的软件缺陷(按曲线估算约有 30 个软件缺陷剩余),这与后续 6 天(补充测试阶段)的实际软件缺陷发现分布(新发现 21 个软件缺陷)基本吻合,说明 ABDOM 初步具备在前期软件缺陷发现数据的基础上对预测后续阶段中软件缺陷发现情况的基本能力.

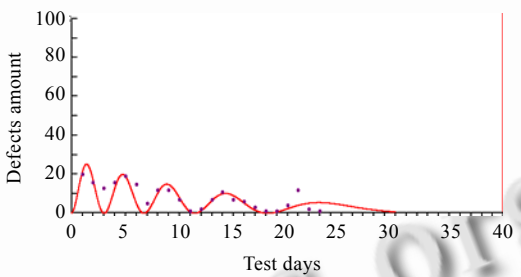

Fig.4 Automatic fitting result of project *B*'s former 17 days defects discovery data by ABDOM

图 4 ABDOM 对某政务信息系统 *B* 测试前 17 天的软件缺陷发现数据的自动拟合结果

4 ABDOM 模型应用讨论

在一个较为规范的测试过程中,ABDOM 模型可以辅助测试管理人员对软件缺陷发现过程进行预测与评价,并进而对被测软件质量进行评价.可以通过 ABDOM 模型对历史数据的拟合,获得该测试组织的软件缺陷发现能力、软件缺陷发现周期、软件缺陷发现阻尼等规律性数据.根据这些历史数据,可以考虑开展 3 种类型的预测或评价:

1) 在测试项目准备期,可以通过预设的软件缺陷发现阻尼、软件新版本发布日程安排、新版本中新增代码量和软件测试预计结束日期等信息对该项目的软件缺陷发现时序过程进行预先估算,并可以评估在预计的时间限制下软件缺陷发现的最终收敛状况如何,进而可评估该测试项目中投入的测试人力资源、时间资源的充分性.

2) 在测试项目执行过程中,在ABDOM模型中输入测试前期获得的软件缺陷发现时序数据,通过拟合来获得该项目中实际的软件缺陷发现能力A_{max}、软件缺陷发现周期τ、软件缺陷发现阻尼a、软件缺陷

发现周期阻尼b等参数.在此基础上,可通过ABDOM模型给出的测试后期的软件缺陷发现时序曲线来预测后期的软件缺陷发现状况;同时,若拟合结果中A_{max}明显低于预期值,软件缺陷发现阻尼系数a也偏低(例如低于 0.1),则可以考虑存在有前期测试过程不规范、各软件缺陷发现周期中测试深度不够的问题.

3) 在测试项目结束后,可以使用 ABDOM 模型来拟合整个软件缺陷发现时序过程,若拟合结果中的软件缺陷发现阻尼系数 a 偏低(例如低于 0.1),同时软件缺陷发现周期阻尼系数 b 偏低(例如低于 0.05),则可初步判断被测软件系统中仍然剩余较多的软件缺陷,并考虑通过补充测试来提高软件质量.

5 结 论

利用 ABDOM 对真实测试项目历史数据的拟合结果和预测验证结果初步表明,本文提出的 ABDOM 模型可较准确地反映出以发现软件缺陷为主要目标、软件测试过程管理规范的软件测试过程中软件缺陷发现过程的时序性、周期振荡性和衰减性等特征,并可对增量式测试中所呈现的软件缺陷非单调衰减特性进行较好表达.拟合结果表明,模型中各参数的取值较好反映了测试团队缺陷发现能力、软件测试质量、软件版本变更规模和变更周期等因素及其对软件缺陷发现过程的影响.在软件测试管理实践中,基于采集的测试过程基本信息,可利用 ABDOM 模型来开展测试进展和被测软件质量的评估活动.

在后续的研究中,主要将考虑如下几个研究方向:

1) 对该 ABDOM 的离散化改进,同时考察其他类型收敛因子的引入,以表达更复杂的软件缺陷发现场景,并进行实践数据验证;

2) 通过对各种已知测试质量的测试项目数据的拟合实验,对软件缺陷发现阻尼 a 和软件缺陷发现周期阻尼 b 物理意义进行深入探讨,进而讨论参数 a 和 b 的规范化问题;

3) 使用 ABDOM 模型对实际的软件缺陷发现过程进行本文第 4 节中所述 3 种类型的预测和评估,对设想的模型应用进行实践检验与修正;

4) 改进 ABDOM 模型,尝试描述开源社区软件测试中的软件缺陷发现时序规律.

References:

[1] Myers GJ. The Art of Software Testing. Wiley Interscience, 1979.

[2] Gaffney JE. Estimating the number of faults in code. IEEE Trans. on Software Engineering, 1984,10(6):459−464. [doi: 10.1109/TSE.1984.5010260]

[3] Malaiya YK, Denton J. Estimating the number of residual defects. In: Proc. of the 3rd IEEE Int'l High-Assurance Systems Engineering Symp. (HASE'98). 1998.

[4] Malaiya YK, Denton J, Li MN. Estimating the number of defects: A simple and intuitive approach. In: Proc. of the 9th Int'l Symp. on Software Reliability Engineering. 1998. 307−315.

[5] Halstead MH. Elements of Software Science. New York: Elsevier, North-Holland, 1977.

[6] Lyu MR. A phase-based approach to creating highly reliable software. In: Proc. of the 24th Annual Int'l Computer Software and Applications Conf. (COMPSAC). 2000. 276.

[7] Carol S, Martin S. Software reliability modeling: An approach to early reliability prediction. IEEE Trans. on Reliability, 1998,47(3): 268−278. [doi: 10.1109/24.740500]

[8] Briand LC, Emam KE, Freimut BG. A comprehensive evaluation of capture-recapture models for estimating software defect content. IEEE Trans. on Software Engineering, 2000,26(6):518−540. [doi: 10.1109/32.852741]

[9] Zhu YC, Xu H. Empirical-Based software defect content estimation improvement. Journal of Beijing University of Aeronautics and Astronautics, 2003,29(10):947−950 (in Chinese with English abstract).

[10] Cai KY. On the neural network approach in software reliability modeling. Journal of Systems and Software, 2001,58:47−62. [doi: 10.1016/S0164-1212(01)00027-9]

[11] Zhang JH, Sun F, Xie RS, Hao YL. Neural network based prediction of software faults in integrated navigation system. Journal of Harbin Engineering University, 2001,22(1):55−58 (in Chinese with English abstract).

[12] Bai CG. Estimation of the Number of Remaining software errors based on Bayesian network. Computer Engineering, 2003,29(18): 39−40 (in Chinese with English abstract).

[13] Bai CG, Yu MH, Hu SX. Software failure predication model based on multiple Markov Bayesian network. Computer Engineering and Applications, 2003,39(10):40−42 (in Chinese with English abstract).

[14] Bergander T, Luo Y, Hamza AB. Software defects prediction using operating characteristic curves. 1-4244-1500-4/07/, IEEE, 2007.

[15] Liu C, Jin MZ. Software Test Process Model-POCERM. Journal of Beijing University of Aeronautics and Astronautics, 1997,1:56−58 (in Chinese with English abstract).

[16] Fenton N, NeiI M. A critique of software defect prediction models. IEEE Trans. on Software Engineering, 1999,25(5):675−689. [doi: 10.1109/32.815326]

附中文参考文献:

[9] 朱永春,徐红.一种基于历史数据的软件缺陷预测方法改进.北京航空航天大学学报,2003,29(10):947−950.

[11] 张家海,孙枫,谢荣生,郝燕玲.估测组合导航系统软件缺陷的一种神经网络方法.哈尔滨工程大学学报,2001,22(1):55−58.

[12] 白成刚.基于 Bayes 网的软件残留错误数度量.计算机工程,2003,29(18):39−40.

[13] 白成刚.基于多重马尔可夫 Bayes 网的软件失效预测模型.计算机工程与应用,2003,39(10):40−42.

[15] 刘超,金茂忠.软件测试过程的基本模型 POCERM.北京航空航天大学学报,1997,1:56−58.

何智涛(1972－),男,湖北江陵人,博士生,讲师,主要研究领域为软件测试,测试过程管理.

刘超(1958－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为软件测试,软件工程.

晏海华(1964－),男,副教授,主要研究领域为软件测试,软件工程.