基于可靠性增长模型的软件可靠性增长测试充分性准则*)

吴彩华[1] 朱小冬[1] 刘俊涛[2] 王毅刚[1]
(军械工程学院维修工程实验中心[1] 计算机工程系[2] 石家庄 050003)

摘 要 结合软件可靠性增长模型,扩充了基于可信度度量的软件可靠性增长测试充分性问题度量准则,提出了关于可靠性增长测试充分性问题新的度量准则:可靠性测试只有同时满足可靠性增量指标要求、可信度要求和每单位测试资源发现的故障数要求方可终止。然后,以一个真实数据集为例,应用所提出度量准则求出了测试应该停止的时间。实验证明,这样的度量准则,使得终止条件更加严格,可靠性测试更加充分,为可靠性测试充分性问题的解决提供了新的思路。

关键词 可靠性测试,可靠性增长模型,可信度,测试资源

Adequacy Criteria of Software Reliability Testing Based on SRGMs

WU Cai-hua[1] ZHU Xiao-dong[1] LIU Jun-tao[2] WANG Yi-gang[1]
(Maintenance Engineering Institute, Department six[1], Sect. Software Engineering, Department five[2], Mechanism Engineering College, Shijiazhuang 050003, China)

Abstract Combining the software reliability growth model, extended the adequacy criteria of software reliability growth testing based on reliability measurement, presented a new adequacy criteria: reliability testing can't be ended until it satisfies the requirement of reliability increment index, confidence and fault number per testing cost. And then it took a true data set for an example, got the ending time by using the new adequacy criteria. It was proved by the experiment that the new criteria makes the ending condition more rigorous and reliability testing more adequate, and it provides a new thinking for solving the reliability testing adequacy problem.

Keywords Reliability testing, SRGMs, Confidence, Testing resource

1 引言

进行软件测试时,理想的情况是对输入空间中的所有测试数据进行测试,以排除所有的错误。但这几乎是不可能实现的,因为即使对一个非常简单的软件来说,测试所有可能的输入数据所耗费的测试资源也是难以承受的,排除所有的错误也是不现实的。所以引出的一个重要的问题是何时停止测试,使得软件在已经执行的测试数据上的表现能够代表软件的整体表现。这就是软件测试充分性问题需要研究和解决的问题。"测试充分性"达到一定要求时,测试就可以停止,所以有时也称之为测试终止规则。

软件测试充分性最初是由 Goodenough 和 Gerhert 在验证程序正确性时提出的[1]。软件测试充分性是指根据软件在有限多个测试数据上的行为能够推断出软件在所有输入数据上的行为[1]。需要说明的是,软件测试充分性与软件的正确性或软件可靠性之间没有必然联系,因为软件测试充分性只是对测试数据充分性的一种度量,软件可靠性测试是保障软件质量和可靠性的重要手段,同软件测试一样,也面临充分性问题。软件可靠性测试充分性问题包含于软件测试充分性问题,但不能说软件可靠性测试充分性问题等同于软件测试充分性问题。当前,对软件测试充分性的研究大多集中于基于程序结构的控制流和数据流的测试充分性上,常以某覆盖项(如语句、分支、路径等)的覆盖率达到某一预定的要求作为终止测试的标准。而对软件可靠性测试充分性问题研究则起步较晚,主要集中在满足可靠性指标的要求作为终止测试的标准。文献[2]在国内首先提出了软件可靠性测试充分性问题,给出了软件测试充分性准则的概念,分类以及与整个软件测试充分性准则的关系,提出了满足可靠性指标要求的可靠性验证测试终止准则;但没有对可靠性指标的可信度做要求。文献[3]细化了可靠性测试充分性准则,将可靠性测试充分性准则分为可靠性验证测试充分性准则和可靠性增长测试充分性准则。提出可靠性验证测试充分性准则是判断样本的统计特征与整体的统计特征的接近程度;遗憾的是这两种可靠性测试充分性准则都没有考虑测试资源消耗情况在可靠性测试中的作用。实际上,通过每单位测试资源所发现的错误数的变化情况可以推断出测试是否要终止。文献[4]利用基于可信度的统计覆盖测试技术,提出了一种新的软件可靠性验证测试充分性准则,并利用假设检验理论,求出了软件在不允许失效情况下,所需测试用例数量的最小值,并给出了求可信度的计算公式;但没有给出在允许失效的情况下,所需的测试用例数量。实际上,微量的、不妨碍软件正常运行的错误是可以接受的,如果完全不允许出错,那么按照该方法求出的测试用例的数量是惊人的。软件可靠性验证测试的目的是验证软件的可靠性指标是否达到预定要求,所以仅仅考虑测试数据数量的要求是不够的,更主要的是使可靠性指标达到要求。在以前学者的研究中,大多提出的是可靠性验证充分性准则,只

*)本文受十一五国防预先研究项目(软件密集型装备保障技术,513270104)资助。吴彩华 博士研究生;朱小冬 教授,博士生导师;刘俊涛 讲师;王毅刚 讲师,博士。

是在软件要交付用户使用前而进行的验收测试中使用。而在实际应用中，可靠性增长测试应用更为广泛，所以本文致力于可靠性增长测试充分性准则。本文针对文献[3]没有考虑可靠性指标的可信度的缺点，在度量准则中加入了可信度的约束；针对文献[4]没有考虑测试资源对测试充分性的影响，在度量准则中加入了单位测试资源所发现的错误数的约束；基于以上改进，本文提出了一个新的软件可靠性增长测试充分性准则：可靠性测试只有同时满足可靠性增量指标要求及其可信度要求、单位测试资源排除错误数要求方可终止。

在本文的第 2 部分，提出了新的可靠性增长测试充分性度量准则；第 3 部分应用一个具体的实例对提出的充分性度量准则进行了验证，最后总结全文，并指出了其不足以及今后努力的方向。

2 软件可靠性增长测试充分性度量准则

本文在分析了文献[3]中提出的满足可靠性指标要求的可靠性测试终止准则基础上，扩充了基于可信度度量的软件可靠性增长测试充分性度量准则，结合可靠性增长模型，提出了关于可靠性增长测试充分性新的度量准则：可靠性测试只有同时满足可靠性增量指标要求、模型可信度要求以及单位测试资源所发现的故障数要求方可终止。

2.1 可靠性增量指标要求

常用的软件可靠性指标有可靠度、失效率、MTTF、MTBF 等。本文以可靠度为例，要想终止测试，规定估计的可靠度增量必须满足如下要求：

$$\Delta R_i < R_m \tag{1}$$

其中 $\Delta R_i = R_i - R_{i-1}$，$i=1,2,3\cdots$，$R_i$ 为软件的可靠度，可根据建立的可靠性增长模型估计得到，R_m 为一个接近 0 的正数。当 ΔR_i 达到 R_m 时，我们认为软件的可靠度在经过测试-修改的过程后增长得很小，测试已经不能产生明显效果，这时可以考虑停止测试了。

2.2 可靠性模型的可信度要求

要想终止测试，所用的可靠性增长模型的可信度必须满足如下要求：

$$C_i \geqslant C_m \tag{2}$$

C_i 为 t_i 时刻可靠性增长模型的可信度，我们采用如下公式计算：

$$C_i = 1 - \frac{1}{i}\sum_{1\leqslant j\leqslant i}\left|\frac{y_j - m(t_j)}{y_j}\right| \tag{3}$$

其中 $m(t_j)$ 表示在 t_i 时刻用可靠性增长模型估计的 t_j 时刻的累积故障数，y_j 表示在 t_j 时刻实际观测的累积故障数。C_i 实际上表示了用可靠性模型估计的值与实际值的吻合程度，我们用其来评估所用可靠性模型的可信度。如果所用可靠性模型的可信度符合要求，我们就认为用此模型预测得到的软件的可靠度是可信的，与此同时，如果估计的软件可靠度也不再增长(或增长得足够少)，则可考虑停止软件测试。

2.3 测试资源的约束

在软件测试中，单位测试资源排除故障数反映了测试团队的测试效率，也是一个重要指标。当单位测试资源排除故障数比较大时，略微增加测试资源即可发现更多的错误，因此应该继续进行软件测试；另一方面，当单位测试资源排除故障数比较小时，即使投入更多的测试资源也难以产生很好的效果，此时应该考虑终止测试。有鉴于此，需要加入单位测试资源所发现的故障数的约束，即当满足式(4)时，应该考虑停止测试。

$$\frac{\Delta m(t_i)}{\Delta W(t_i)} = \frac{m(t_i) - m(t_{i-1})}{W(t_i) - W(t_{i-1})} \leqslant d \tag{4}$$

其中，$m(t_i)$为 t_i 时刻估计的累积故障数，$W(t_i)$为 t_i 时刻累积消耗的测试资源，d 为给定的单位测试资源所发现的最少故障数。

当软件测试进行到一定阶段时，如果用可靠性增长模型预测的结果同时满足了(1)、(2)、(4)式时，测试就可以终止了。也就是说，此时如果再进行进一步的测试则不能产生很好的效果，并浪费测试资源；如果测试在此之前停止则会遗留更多的软件故障。

3 实例及讨论

下面通过一个实例进一步说明我们提出的充分性准则。本例采用 1984 年 Ohba 收集的数据集[5]，它是对一个 PL/I 数据库应用软件进行测试的结果。该软件包括 1317000 行代码，在开始的 19 周时间内，花费 47.65CPU 时间共排除了 328 个错误。在测试了很长时间后总的累积故障数为 358。具体数据参看表 1。利用这些数据，我们以测试进行到第 9，15，18 周时为例，应用提出的充分性准则来判断何时应该停止测试。

需要指出的是如果测试到第 i 周(本例中 $i\leqslant 19$)预测测试何时终止时，应该以第 i 周之前的数据为参考，也就是说，随着测试的进行，我们要依据新得到的数据对以前预计的结果进行修正；因此是一个不断测试、不断预测的过程，这样得出的结果才更准确。

表 1 PL/I 程序数据集

观察时间(周)	累积执行时间	累积故障数	观察时间(周)	累积执行时间	累积故障数
1	2.45	15	11	26.23	233
2	4.90	44	12	27.67	255
3	6.86	66	13	30.93	276
4	7.84	103	14	34.77	298
5	9.52	105	15	38.61	304
6	12.89	110	16	40.91	311
7	17.10	146	17	42.67	320
8	20.47	175	18	44.66	325
9	21.43	179	19	47.65	328
10	23.35	206			

3.1 可靠性增长模型的建立及求解

可靠性增长模型的建立对可靠性评估精度的影响很大，所以，要对失效数据进行严密的分析来选取适合的可靠性增长模型。在本文中，为了估计测试所用的测试资源，选用了文献[6]提出的可靠性增长模型：该模型是在传统的 GO 模型的基础上，引入了测试资源函数 $w(t)$，即

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}\times\frac{1}{w(t)} = r(a - m(t)) \tag{5}$$

其中，$m(t)$表示在 t 时刻的累积故障数，$w(t)$表示 t 时刻的测试资源消耗，r 为检错率，a 为初始故障数。

解(5)，得出：

$$m(t) = a(1 - \exp(-r(W(t) - W(0)))) \tag{6}$$

$$R(\Delta t \mid t) = \exp[-m(t+\Delta t) - m(t)] \tag{7}$$

其中，$R(\Delta t \mid t)$为 t 时刻可靠度，表示软件在 t 时刻出错的情况下，在$[t, t+\Delta t]$内不出错的概率。$W(t) = \int_0^t w(\tau)\mathrm{d}\tau$，在本

文中，采用文献[6]C. Y Huang 提出的 $W(t)$ 的估计方法

$$W(t)=\frac{N}{\sqrt[k]{1+A\exp[-\alpha kt]}} \tag{8}$$

根据文献[6]，它的各个参数如表 2 所示。

表 2　测试资源函数各个参数的值

N	A	α	k
48.7768	429.673	0.1580	2.63326

我们采用最大似然法估计模型中的参数。由于选用数据过少造成估计结果不准确，因此，本文首先选用前 9 组数据（即先测试 9 周）对可靠性模型的参数进行估计，在第 9，15，18 周对模型参数估计的结果如表 3 所示。

表 3　在 9，15，18 周估计的模型参数值

预测时间(周)	估计参数值
9	a=434.2131；r=0.0363
15	a=388.7596；r=0.0421
18	a=429.6858；r=0.0368

3.2　测试充分性判定

根据上面模型求解的结果，可以分别计算出估计累积故障数 $m(t_i)$，$R(0.01\mid t_i)$，估计测试资源，进而可求出 $\Delta R(0.01\mid t_i)$、模型可信度 C_i 和单位测试资源排除故障数。在第 9 周进行测试充分性判定的计算结果如表 4 所示。

对于这个软件，假设要求的终止条件为：

$$\begin{cases} C_c\geqslant 0.8 \\ \dfrac{\Delta m(t_i)}{\Delta w(t_i)}\leqslant 3.3 \\ \Delta R(0.01\mid t)\leqslant 0.002 \end{cases} \tag{9}$$

在第 9 周模型的可信度 $C_9=0.8048$，满足了对模型可信度的要求。从表 4 中可以看到，在第 24 周时，$\Delta R(0.01\mid t_i)$ 和单位测试资源排除故障数已经满足了式(9)，因此预测测试可以在 24 周结束。如果根据文献[2]提出的充分性准则，仅仅只考虑可靠性指标的要求，例如要求 $R(0.01\mid t)\geqslant 0.95$，则测试到第 18 周时就应该终止，而实际上在第 18 周结束后还有 33 个错误没有被测试到，并且在接下来的一周中还能找到 3 个错误。而将测试进行下去也不会额外的花费过多的测试资源。因此，本文提出的测试充分性准则在同时提高测试效果和节省测试资源方面优于文献[2]提出的准则。

同样的方法，我们可以在第 15，18 周利用前面估计的模型参数计算 $\Delta R(0.01\mid t_i)$、模型可信度 C_i 和单位测试资源排除故障数这些指标，并按照式(9)进行充分性判定。图 1 显示了在 9，15，18 周对累积故障数的预测结果与真实情况的对比。表 5 显示了在 9，15，18 周时模型的可信度，它们均满足可信度要求。图 2 显示了在 9，15，18 周时对 $\Delta R(0.01\mid t_i)$ 的估计情况，我们可以看到第 15 周时预测第 23 周 $\Delta R(0.01\mid t_i)$ 将满足要求，第 18 周时预测第 24 周 $\Delta R(0.01\mid t_i)$ 将满足要求。图 3 显示了在 9，15，18 周时对 $\frac{\Delta m(t_i)}{\Delta w(t_i)}$ 的估计情况，我们可以看到第 15 周时预测第 17 周 $\frac{\Delta m(t_i)}{\Delta w(t_i)}$ 将满足要求，第 18 周时预测第 23 周 $\frac{\Delta m(t_i)}{\Delta w(t_i)}$ 将满足要求。因此，我们在第 15 周预测第 23 周将终止测试；在第 18 周预测第 24 周将终止测试。在各个预测时间点预测的测试终止时间，如表 6 所示。

表 4　在第 9 周估计的结果

观察时间(周)	实际累积故障数 m_i	估计累积故障数 $m(t_i)$	$R(0.01\mid t_i)$	$\Delta R(0.01\mid t_i)$	估计累积测试资源 $W(t_i)$	单位测试资源排除故障数
10	206	204.9265	0.7733	0.0094	22.4652	7.8572
11	233	230.1113	0.7827	0.0167	25.6705	6.9733
12	255	253.6364	0.7994	0.0232	29.0441	6.1654
13	276	274.6587	0.8226	0.0276	32.4538	5.4603
14	298	292.5690	0.8503	0.0292	35.7339	4.8730
15	304	307.1077	0.8795	0.0277	38.7174	4.0154
16	311	317.3824	0.9072	0.0241	41.2762	4.5325
17	320	326.7805	0.9313	0.0194	43.3497	3.7890
18	325	332.8319	0.9507	0.0148	44.9468	3.6024
19	328	337.0809	0.9655	0.0108	46.1263	3.4724
20		340.0078	0.9763	0.0076	46.9692	3.3837
21		341.9964	0.9839	0.0053	47.5569	3.3231
22		343.3346	0.9892	0.0036	47.9596	3.2817
23		344.2292	0.9928	0.0024	48.2322	3.2553
24		344.8246	0.9952	0.0016	48.4151	3.2385
25		345.2197	0.9968	0.0011	48.5371	3.2269
26		345.4814	0.9979		48.6182	

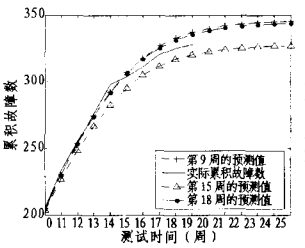

图 1　预测累积故障数与实际累积故障数

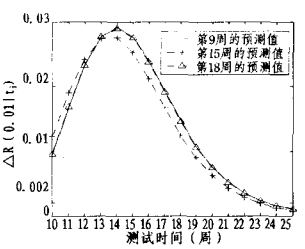

图 2　预测的 $\Delta R(0.01\mid t_i)$ 值

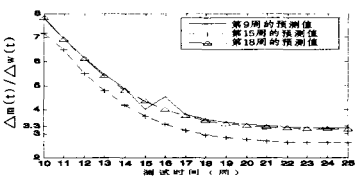

图 3　预测的 $\Delta m(t_i)/\Delta w(t_i)$ 值

表 5　在 9，15，18 周模型的可信度

预测时间(周)	模型可信度 C_i
9	0.8048
15	0.88
18	0.8965

表 6　在各个预测时间预测的测试终止时间

预测时间(周)	预测终止时间(周)
9	24
15	23
18	24

结束语　本文提出了基于可靠性增长模型的软件可靠性测试充分性问题新的度量准则：可靠性测试只有同时满足可靠性增量指标要求及其可信度要求、每单位测试资源发现的故障数要求方可终止。与其它可靠性测试充分性度量准则相比，该度量准则考虑了测试资源对可靠性测试充分性的影响，因此，它更接近实际情况；但是，该度量准则也具有一定的局限性，比如它没有考虑软件的规模，软件的内部结构等因素，所以，在今后的研究中，要和软件测试充分性度量标准相结合考虑。

（下转第 292 页）

由此可见，我们设计的 HUNTBot 的结构具有以下特点：(1)灵活性。混合式结构保证了 NPC 既能对环境迅速做出反应，又能对目标进行规划；(2)具有社会性。NPC 的信念集中有对盟友的认知，决策时受到团队协议的影响，并能与盟友进行通信。(3)学习能力。决策模块中的学习器保证了 NPC 具有对行为和规划进行学习的能力。(4)预测能力。预测模块使 NPC 具有对未来世界状态的认知，从而可以辅助其决策。

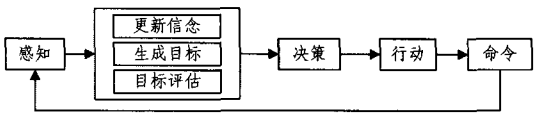

图 3　HUNTBot 的处理流程

3.2　HUNTBot 的处理流程

HUNTBot 的一个完整的处理流程如图 3 所示，包含以下 5 个步骤：

Step1 感知。NPC 接收感知器发来的各种信息。其中包含视觉信息、听觉信息，以及自身的状态，如武器装备、生命值、位置、方向、速度等等。

Step2 信念更新、生成目标、目标评估。包含以下几个部分：

①信念更新。将感知阶段接收到的信息按类别加入信念集中，对当前信念集进行更新。

②生成目标。加入新的信念可能产生任务相关的数据(目标)。例如，加入“看到有用武器”的信念，会产生“获得武器”这一目标。

③目标评估。信念集内可能产生多个目标。例如，“获得武器”和“躲避攻击”目标同时出现。此时需要为候选目标排序。如“躲避攻击”优先级高于“获得武器”。

Step3 决策。为信念集中优先级最高的目标(如“躲避攻击”)进行规划，产生动作序列。如“找到隐蔽处—到达隐蔽处—蹲下”。若此时有紧急事件需要处理，则触发反应式动作。

Step4 行动。将决策产生的动作序列按照相互之间的制约关系排序，以使动作协调运行。

Step5 命令。将排好序的动作以 Gamebots2004 认可的格式输出。

3.3　HUNTBot 的实验例

图 4　HUNTBot 的社会性

我们在 Pogamut 2 上实现了 HUNTBot，运行结果显示使用其构建的 NPC 兼具反应性和思考性，并具备社会性和学习能力。图 4 显示当 NPC 收到其他盟友发送的消息，导致信念集中出现 SquadAssmble(集合队伍)这一信念时，立即触发其反应式行为生成命令 run(跑动)，跑向队伍集合地。图 5 显示当 Agent 发现敌人时，具有目标 EliminateEnemy(消灭敌人)，由于曾经学习到完成这一目标所规划的行动序列，因此直接执行命令序列中包含的 Goto(跑至某一点)和 Fire(射击)。

图 5　HUNTBot 的学习能力

结束语　大型的电脑游戏要求 NPC 具备更加智能的行为，从而很好地适应环境的变化。我们针对 FPS 游戏——“虚幻竞技场”设计了一种 NPC 结构——HUNTBot。在 Pogamut 2 平台上的初步实验表明，HUNTBot 既能对环境变化作出即时反应，又能根据目标进行长期规划，并具有团队合作和学习能力，在游戏环境中表现出较好的性能。在今后的研究中，我们会考虑将实时规划技术引入 HUNTBot，使规划过程更适应实时性要求较高的游戏环境，并考虑采用强化学习技术对团队行为进行学习，使 NPC 的智能行为更加接近人类玩家。

参 考 文 献

[1] Champandard A J. AI game development. Indianapolis: New Riders Press, 2003

[2] Nareyek A. AI in Computer Games. ACM Queue, 2004, 1(10): 58-65

[3] Magerko B, Laird J E, Assanie M, et al. AI Characters and Directors for Interactive Computer Games. AAAI, 2004: 877-883

[4] Orkin J. Agent Architecture Considerations for Real-time Planning in Games. AIIDE, 2005: 105-111

[5] Burkert O, Kadlec R, Gemrot J, et al. Towards Fast Prototyping of IVAs Behavior: Pogamut 2. IVA, 2007: 362-363

[6] 王皓，罗文杰，高阳，等. 一种基于 Agent 的智能游戏开发平台及其 NPC 设计. 计算机科学，2007，34(9A)：66-69

[7] Kaminka G A, Veloso M M, Schaffer S, et al. GameBots: a flexible test bed for multiagent team research. Commun. ACM, 2002, 45(1): 43-45

[8] Champandard A J. Flexible embodied agent architecture. http://sourceforge.net/projects/fear/

(上接第 283 页)

参 考 文 献

[1] Goodenough J B, Gerhert S L. Toward a theory of test data selection[J]. IEEE Transactions on Software Engineering, 1975 (SE-3): 156-173

[2] 李秋英，陆民燕，阮镰. 软件可靠性测试充分性问题的理论研究. 北京航空航天大学学报，2003，29(4)：312-316

[3] 李秋英，阮镰，刘斌. 软件可靠性测试充分性研究. 测控技术，2003，22(11)：49-52

[4] 沈升源，陈丽容，汤铭端. 基于统计覆盖测试技术的软件测试充分性研究. 系统工程与电子科技，2004，26(6)

[5] Ohba M. Software reliability analysis models. IBM Journal of Research and Development, 1984, 28(4): 428-443

[6] Huang Chin - Yu. Performance analysis of software reliability growth models with testing-effort and change-point. The Journal of Systems and Software, 2005, 76: 181-194