突变是测试实验的合适工具吗？

J.H.安德鲁斯
计算机科学系
西安大略大学
加拿大伦敦
andrews@csd.uwo.ca

L.C.布里安德Y。拉皮切
软件质量工程实验室
系统及计算机工程系
卡尔顿大学
加拿大渥太华
{briand, labiche}@sce.卡尔顿。ca

摘要
测试技术的实证评价在软件测试研究中起着重要的作用。一种常见的做法是检测故障，可以手动或使用突变操作符。后者允许对大量故障进行系统的、可重复的播种；然而，我们不知道通过这种方法获得的实证结果是否会导致有效的、有代表性的结论。本文基于一些具有综合测试用例和已知故障的程序来研究这一重要问题。结论是，基于目前可用的数据，使用突变操作符可以产生可信的结果（生成的突变与真实的错误相似）。然而，突变体似乎不同于手工种子的错误，后者似乎比真正的错误更难被发现。

类别和主题描述符
.2.D5测试和调试

一般条款
实验、验证。

关键字
真正的错误，手工播种的错误，突变体

1.介绍
实验是软件测试研究的重要组成部分。通常，实验是用来确定两种或以上方法中哪一种优于执行一些测试相关的活动。例如，人们可能会对比较用于推导测试用例的几个测试标准的故障检测有效性感兴趣，并诉诸于为此目的的实验。测试实验通常需要一组有已知故障的主题程序。这些主题程序应该足够大，以现实，但不能大到使实验不可行。对于故障，一种技术处理给定故障的能力应该能够准确地预测该技术在实际程序上的性能。

测试实验设计中的一个问题是，很难找到适当大小的真实错误的真实程序，也很难适当地准备（例如，通过准备正确和错误的版本）。即使有实际故障的实际程序可用，这些故障通常也不足以使实验结果达到统计意义。因此，许多研究人员采用了将错误引入正确的程序的方法，以产生错误的版本。

这些故障可以通过手动引入（例如，实验人员可以要求有经验的工程师这样做），或者通过自动生成代码的变体。通常，我们将自动生成的变体视为对代码应用操作符的结果。以这种方式使用的操作符称为突变操作符，由此产生的错误版本称为突变体，一般的技术称为突变体或突变体生成。

突变体产生的主要潜在优势是，突变体操作符可以被精确地描述，从而提供一个有明确定义的错误播种过程。这有助于研究人员复制他人的实验，这是良好实验科学的必要条件。虽然手工引入的错误可以被认为更现实，但最终它是一个特定错误是否现实的主观判断。突变体产生的另一个重要优势是，可能会产生大量的突变体，这增加了所获得的结果的统计意义。

然而，一个重要的问题仍然存在。我们如何知道检测（或“杀死”）突变体的能力是否是实际表现的准确预测器；也就是说，突变体在评估测试技术的故障检测有效性时的外部有效性是什么？这个问题的答案将对我们如何进行测试实验，从而对实验结果的有效性产生重要的影响。

本文的主要贡献是比较了测试套件对手工种子故障、自动生成故障和真实故障的故障检测能力。我们的主题程序是一套被广泛使用的具有手工种子故障的程序，以及一套被广泛使用的具有真实故障的程序。我们使用来自关于突变测试的文献中的一组标准突变操作符从主题程序中生成突变体。结果表明，所产生的突变体与真实故障相似，但不同于手工种子故障，且手工种子故障比真实故障更难检测。

2. 相关工作

使用突变体来测量测试套件的充分性的想法最初是由DeMillo等人提出的。[6]和Hammet[12]，并由Offutt和其他人[20]广泛探索。Offutt[17]显示了对突变测试的一个基本前提的经验支持，即检测简单故障的测试数据集
（比如那些由突变引入的故障）将检测到复杂的故障。e.，几个简单故障的组合。

弗兰克尔和Weiss[10]，瑟韦诺德-福斯等人已经使用错误的变体程序进行了实验。[24]和哈钦斯等人。[14]，此后被许多研究人员研究。弗兰克尔和韦斯使用了9个帕斯卡程序，每个程序都有一个现有的故障，而哈钦斯等人。在所使用的7个程序中，手工播种了130个故障。Thevenod-Fosse等人。使用突变操作符自动在四个小的C程序中播种故障。一般来说，这些实验遵循的模式生成一个大的“测试池”的测试用例，运行所有错误版本的测试用例，观察哪些测试用例检测故障，并使用这些数据推断出故障检测能力给定的测试套件从池(e。g.，满足特定覆盖标准的测试套件)。

虽然突变体产生最初是作为测试策略的一部分，注意Thevenod-Fosse等人。用它作为一种为实验生成错误版本的方法。其他这样做的研究人员包括Kim等人。[15]，Memon等人。[16]，安德鲁斯和张[1]，Briand和Labiche[2]。最后，陈等人。[5]在他们的实验中使用了手工种子的错误和产生的突变体。他们指出，手工播种的故障只是可能的故障的一个子集，并提出了故障是否具有代表性的问题，但由于这不是他们研究的重点，他们没有进一步探讨它。
总之，据我们所知，还没有实证研究通过将突变体或手工种子故障与真实故障的结果进行比较来直接评估它们的使用。

3. 实验材料和程序

在本节中，我们将描述我们所执行的实验，主题程序和与之相关的工件（错误版本和测试用例），以及我们用于产生突变体的突变操作符。

在本文的其余部分中，我们使用术语手工种子错误(e。g.，由一个经验丰富的工程师产生的)和真正的故障(e.g.，在软件开发过程中发现的)，或者仅仅是在没有歧义时出现的错误，而不是由被称为突变体的突变操作符产生的错误。

表1总结出了该主题程序的属性。“NLOC”是代码行，i。e.，删除注释后的非空白的代码行。“#条件条件”是C条件构造的数量（如果，同时，情况等）。以及二进制逻辑运算符（&&和||），包括作为代码复杂性的粗略指标。

3.1实验的定义

为了实现引言中所述的目标，我们分析了测试套件的检测率。我们使用了8个主题程序（表1），其中故障和大量的测试用例池可用：注意，这些测试池具有可比性，因为它们确保满足了几个结构标准（第3.2节）。然后，通过对每个主题程序的大型测试池进行随机抽样，可以形成测试套件（第3.4节）。然后，我们创建这些程序的突变版本（第3.3节），并对可用的错误版本和突变版本执行这些测试套件，记录被杀死的故障，以计算测试套件的故障检测比率。这些比率在突变体和错误之间进行比较，以确定它们是否相似。因为每个测试套件可能产生不同的检测比，我们为每个主题程序获得了两个分布，分别为故障和突变体。通过统计推理检验，比较了这些分布的中心趋势。零假设(H_0)的表述如下：故障集和突变体集之间的检测率没有差异。另一种假设(H_a)表示故障和突变体之间的检测率存在差异。断层分布和突变体分布之间的差异必须根据现有的数据来解释，而且还需要调查不同主题项目之间的结果的可能差异。

.23个主题项目

我们使用了一组用C语编写的8个著名的主题程序（表1）。第一个项目是通常被称为“空间”的项目，由欧洲航天局开发，由Frankl等人[9,25]首次用于测试战略评估目的。最后七个是所谓的西门子程序套件与手工种子故障，首先由哈钦斯等人使用。使用[14]来比较基于控制流和基于数据流的覆盖标准。所有8个程序的工件（包括错误版本和测试用例）随后都被其他研究人员修改和扩展，特别是罗瑟梅尔和哈罗尔德[23]和Graves等人。[11].我们选择这些项目是因为那些成熟的项目

表1。主题项目的描述

	主题程序							
准则	空间	打印令牌	印刷机2	更换	进度表	附表2	Tca	Totinfo
nloc	5905	343	355	513	296	263	137	281
#条件	635	94	81	99	37	51	25	46
测试池大小（测试用例号）	13585	4130	4115	5542	2650	2710	1608	1052
故障数量（版本）	38	7	10	32	9	10	41	23
编译突变体的数量	11379	582	375	666	253	299	291	516

相关的人工制品，并因为它们的历史意义：Do等人。[8]报告说，在过去的十年里，有17篇高调的实验软件工程论文使用了西门子套件和/或SorSpace。

太空与38个错误的版本（真正的错误）有关，33个来自最初的沃科洛斯和弗兰克尔研究，5个后来被其他研究人员发现。每个错误版本对应于一个“在测试和程序的操作使用期间”被纠正的错误[25]。已纠正所有错误的“黄金”版本被视为产生正确的输出。相比之下，西门子套件程序的错误版本是手工制作的，“由十个不同的人，大多不知道彼此的工作；他们的目标是产生尽可能现实的错误”[14]。

每个受试者程序的测试池（所有测试用例集）是由原始研究人员和随后的研究人员遵循各种功能（黑盒）测试技术和结构测试覆盖标准构建的。更难等。[13]给出了这些测试池建设的全面历史。西门子程序的测试池开发如下：使用黑盒技术类别分区[21]创建第一组测试用例，然后根据几个结构覆盖标准（所有语句、所有边、所有定义-使用对）进行扩展。除了原始测试集的构建外，都是通过随机输入选择生成的。

3.3突变操作员

为了生成主题程序的突变体，我们使用了一个由Andrews和Zhang[1]首先使用的突变体生成程序，为用C编写的代码生成突变体。为了从源文件中生成突变体，每一行代码都按顺序考虑，并应用四类“突变操作符”中的每一类（只要可能）。换句话说，每个突变操作符对一行代码的有效应用都会生成另一个突变体。这四类突变运算符为：

- -将一个整数常量C替换为0、1、-1、((C)+1)或((C)-1)。
- -用来自同一类的其他运算符替换算术运算、关系运算、逻辑运算、位逻辑运算、递增/递减运算或算术赋值运算符。
- -在一个如果或当的声明中否定该决定。
- -删除语句。

前三个操作符类都来自于Offutt等人。的研究[18]关于确定一组“充分的”突变算子，i。e.，一组操作符S，这样，杀死由S形成的突变体的测试套件往往会杀死由一个非常广泛的操作符类形成的突变体。他们被改编了，这样他们就可以从事C项目，而不是最初研究的形式。第四个操作符也出现在[18]中，因为原始研究[1]的主题程序，期间首次使用突变生成程序，包含大量的指针操作和字段分配语句，不会受到任何足够的突变操作符的影响。

约8.4%的突变体没有编译。所编译的每个主题程序的突变体数见表1。对于空间程序，产生了突变体，在测试套件上运行它们是不可行的。

因此，我们每10个月运行一次测试套件th突变体生成。因为每行产生的突变体数量不遵循任何与每10个的选择相互作用的模式th突变体（这仅取决于细胞系上的结构），这相当于10%的突变体，从所有可能的突变体的均匀分布中随机选择。此外，这还确保了整个源代码中都出现了错误（而不仅仅是几个函数/过程）。

.43分析程序

我们在本节中提供对我们使用的分析程序的简要描述和说明。更多的细节将在下一节中介绍，因为我们将报告这些结果。第一步是生成和编译突变体，并在整个测试池中运行所有突变体和错误版本。所有没有被任何测试用例杀死的突变体都被认为与原始程序相同。虽然这可能不是对每个突变体都适用，但它被认为是一个足够好的近似，在任何情况下，它都是处理大量突变体时的唯一选择，因为自动识别等效突变体是一个不可确定的问题[3,6,19]。对于每个受试者程序，通过随机抽样（没有重复）可用的测试，随机形成5000个测试套件

水池我们需要生成大量的测试套件来获得故障/突变检测率的样本分布，这可以很好地接近潜在的理论分布。大样本还增加了统计测试的能力，并促进了其使用，如下所述。随机选择而不是由覆盖标准驱动的选择。g.，结构性的)被认为提供了足够的可变性。要做的一个决定与测试套件的大小有关。我们希望这在我们的分析中是一个常数，以免模糊我们正在分析的趋势：e。g.，更好的故障/突变体检测有效性可能仅仅是由于测试集的大小，而不是由于（可能的）故障和突变体之间的差异。在测试集中使用太少的测试用例会导致较小的故障/突变检出率，从而阻止我们观察到差异，而在测试集中使用太多的测试用例会减少可变性。我们使用不同的大小进行了分析，以确定它将如何影响结果，从10到100个测试用例。由于没有观察到主要差异不同测试套件大小，和大小100导致合理的故障/突变检测率与良好的可变性测试套件（100对应10%的最小测试池-表1），我们决定只报告结果100大小的测试套件。

然后，我们确定了池中的哪个测试用例检测到哪个突变体和故障。接下来，我们计算了所有测试套件的故障检测比，绘制了每个受试者程序的突变体和故障的检测比分布，然后比较了它们的中心趋势。

综上所述，对于每个测试套件S，该过程产生了两个汇总数据：Dm(S)，S检测到的突变体数量，和Df(S)，S检测到的故障数量。给定主题程序的非等效突变数Nm和非等效故障的Nf，我们计算每个测试套件S的突变检测率Am(S)为Dm(S)/Nm，故障检测率Af(S)为Df(S)/Nf。

我们希望首先检验以下的零假设H_0(P)对于每个受试者项目P：Am和Af比率的平均值

因为P也是一样的。为了做到这一点，我们采用了一个标准的统计检验：匹配对t-检验[22]，原因将在第4节中进一步详细说明[1].如果我们拒绝H_0因为断层和突变体分布的均值差异是统计上和实际上的[2]重要的是，那么下一个问题是为什么？此外，如果不同程序之间的结果不一致，我们还需要确定最合理的解释是什么。有许多可能的原因，包括主题程序的特征差异、测试套件和故障被播种的方式。所有主题程序唯一共同的东西是我们生成突变体的方式。

为了解释我们的结果，以及由于在第4节中进一步解释的原因，我们还查看了杀死每个错误和突变体的测试用例的百分比。对于每个突变体M和错误的版本F，我们计算了K(M)。K(F))，即杀死它的测试用例的数量。然后，我们计算出，对于每个突变体M，杀死该突变体的容易程度E(M)作为E(M)=K(M)/T的比值，其中T是该程序的测试用例总数。（我们对每个错误版本进行E(F)的计算。）对于它们，我们得到了一个分布，其中每个观察结果对应一个突变或错误，我们可以比较这些分布的平均值。

在分析方面，我们首先进行了方差分析(anova)来评估程序均值之间的差异的总体统计学意义。如果显著，那么我们将继续使用对独立样本的t检验来比较每个（空间，西门子程序）对的分布平均值。为了做到这一点，我们求助于Bonferroni程序[22]，以确保在进行大量比较时出现I型错误的低风险：在我们的案例中是28次比较[3].因为我们在这里处理较小的样本(样本量是由数量决定的

[1] 当处理大数据集时（根据经验法则，超过100个观测值），这个检验对于偏离配对t检验的正态性假设（配对之间的差异被假设为正态分布）非常稳健，即使在显示极端分布偏态的情况下也是如此。这一点很重要，因为这种违反假设的行为在现实数据集中很常见。还要注意，在我们的情况下，我们将在我们的观测中不会有极端的异常值，因为我们将处理0和1之间的比率，我们将有5000个观测值供我们处理。然而，为了安全起见，我们使用了另一种等效的非参数检验(Wilcoxon匹配配对符号秩检验)来进行复核结果。一般来说，当使用大样本时，匹配对t检验也特别适合，因为它考虑了差异的大小，因此比等效的非参数检验更强大。

[2] 当处理大样本时，即使是微小的平均值差异也会导致统计上显著的结果。然而，同样重要的是要确定一个具有统计学意义的结果是否具有任何实际意义，即如果平均值上的差异足够大，值得考虑。这通常被称为“实际”意义，有时也被称为“临床”意义。我们对每个(Space，西门子程序)对的Am(S)和Af(S)进行比较，从而得到（7*8）/2=14的比较。同样地，比较E(M)和E(F)也会得到14个额外的比较。

[3]

对于突变体和错误，而不是上面的测试套件)，我们需要用一个等效的非参数检验，即曼-惠特尼检验[20,22]。事实上，在小样本上，违反t检验假设会导致第二类错误的增加，即错误地推断出平均值的差异并不显著。

在我们所有的统计检验中，我们使用α=0作为显著性水平。05，这意味着有5%的几率犯第一类错误，或者换句话说，当差异不是这样时，识别出统计上显著的差异的风险为5%。

.53对有效性的威胁

本节讨论了对我们实验有效性的不同类型的威胁，按照降低优先级的顺序：内部，
外部、构造和结论效度[4,26]。

与内部有效性相关的一个问题是，我们重用了在以前的实验中广泛使用的程序。我们正在使用的程序、测试池和错误并不是根据任何特定的标准来选择的，除了它们是准备充分的和历史上重要的。然而，我们不能保证测试池具有相同的检测能力和覆盖率，尽管文献报道的它们似乎表明它们是以类似的方式生成的，而且相当全面，因为它们涵盖了一些主要的结构性覆盖标准，如第3.2节所述。

我们对太空有真正的缺点，但对其他方面，我们只是由经验丰富的工程师精心设计的“现实的”缺点。我们也不能期望这些程序具有相似的复杂性，无论我们如何定义和衡量它。它们实际上在大小（表1）和控制流的复杂性上差异很大。我们还预计我们的研究结果会根据所选择的突变操作符而变化；我们使用的，如第3.3节中讨论的，是根据现有的文献选择的，以便是一个最小但足够的集合。

另一个可能的威胁是，我们报告了所有程序的独特测试套件大小，而不考虑它们的复杂性，以及从测试池中随机选择的测试用例。然而，100的大小和随机选择相结合被认为是足够的（足够高的检测率和测试套件中足够的可变性）。此外，使用了其他测试套件大小，并没有导致不同的结果，因此在这里没有报告。外部效度与我们将实验结果推广到工业实践中的能力有关。虽然我们只有程序空间的真实故障，但在实验中使用的仪器是真实的，工业程序，有真实的故障或故障由经验丰富的工程师手动播种。然而，由于只使用了一个有真实故障的程序，因此在其他已识别出真实故障的主题程序上复制这项研究将是非常重要的。

构造效度涉及到我们定义度量的方式，以及它是否测量了我们真正想要捕获的属性：测试集的检测能力和故障的可检测性。在上面讨论测试集的大小及其构造时，这一点是合理的。

结论的效度与受试者的选择、数据的收集、测量的信度和统计检验的效度有关。这些问题已经在实验的设计中得到了解决，

表2。描述性统计-突变体和故障的检测比率（测试套件大小为100）

主题程序								
斯塔蒂斯蒂c	Space	复函e	打印令牌	印刷机2	Schedule	附表2	塔卡s	Totinfo
突变体-Am(S)								
中位数	0.75	0.93	0.98	0.99	0.96	0.96	0.91	0.99
平均	0.75	0.93	0.97	0.99	0.96	0.96	0.90	0.99
90%	0.78	0.95	0.99	0.99	0.98	0.97	0.94	0.99
75%	0.77	0.94	0.98	0.99	0.97	0.97	0.93	0.99
25%	0.74	0.93	0.97	0.98	0.94	0.95	0.89	0.98
10%	0.72	0.92	0.96	0.98	0.93	0.95	0.86	0.98
最小	0.65	0.88	0.94	0.93	0.91	0.91	0.77	0.94
最高的	0.82	0.98	1	1	0.99	0.99	0.97	1
故障-Af(S)								
中位数	0.76	0.68	0.57	0.90	0.67	0.67	0.76	0.65
平均	0.77	0.67	0.63	0.93	0.66	0.63	0.79	0.89
90%	0.82	0.77	0.86	1	0.78	0.78	1	0.96
75%	0.79	0.74	0.71	1	0.78	0.78	0.95	0.91
25%	0.74	0.61	0.57	0.90	0.56	0.56	0.68	0.87
10%	0.71	0.55	0.29	0.80	0.44	0.44	0.61	0.83
最小	0.53	0.35	0.14	0.60	0.33	0	0.34	0.65
最高的	0.97	0.93	1	1	1	1	1	1

尽管对所有的程序都有手工种子和真正的错误会更好。

我们的分析和解释将必须考虑到上述所有的因素，以确保所有合理的解释都被考虑在内。

4.分析结果

按照第3.4节中描述的程序，我们首先比较测试套件的检测率(第4节。1).在第4.2节中，我们确定了可能可以解释在第4节中观察到的趋势的现象。1.*为了调查最合理的解释，我们然后比较了在不同程序中杀死错误和突变体的测试用例的百分比，从而观察它们的可检测性，i*。e.，它们易于检测（第4.3和4.4节）。我们将讨论我们的结果的含义（第4.5节）。

4.1突变体与故障的检测分布的比较

我们分析的第一步是比较每个受试者程序的突变体和故障检测比率Am和Af的分布。目的是确定它们之间是否存在统计上和实际上的显著差异。我们在表2中提供了每个主题程序的这些分布的描述性统计数据（具有二进制精度）。请注意，这些是我们在所选测试套件时获得的结果

大小是100。在其他测试套件大小的（10,20,50）中也获得了类似的结果，因此在这里没有报告。

从表2中，我们可以很容易地观察到，一般来说，突变体往往比错误更容易检测。这一趋势的唯一例外是空间，在突变体和断层分布之间没有观察到实际上的显著差异(e。g.，它们的中位数分别为0.75和0.76)。虽然由于空间的限制，并不是所有的分布都可以在这里显示，但图1和图2显示了空间和替换程序的分布直方图，以及标准的分位数盒图，以可视化分布的主要描述性统计数据。箱形图显示了中位数、平均值、各种分位数（1%、10%、25%、75%、90%、99%），以及最小值和最大值。我们现在必须检查一下差异是否有统计学意义。为此，我们执行了一个匹配的配对t-检验[7,22]。

回想一下，在这些分布中，我们比较了5000个测试套件的突变体(Am)和故障(Af)检测率。在每个分布中，观察结果很可能相关，因为它们是成对的。e.，每个Am和Af分布中的一个观察结果对应于同一个测试套件。这就是为什么需要这样的配对测试，而不是一个独立的样本测试。

表3显示了，对于每个受试者程序，t检验的结果，即对间故障/突变检测比的平均差异、t比和p值（这是一个双向检验，因为我们不能预先确定差异的方向）。结果表明，所有p值均接近或等于0，所有差异均具有统计学意义。这甚至就是太空的情况

虽然平均差异为1%，但这实际上并不显著，因为没有测试技术评估会基于如此小的差异得出不同的结论。对于其余的受试者项目，平均差异从6%到34%，平均为22%。与空间相反，这实际上意义重大，因为基于这些范围，如果使用突变体来评估测试技术，它在检测故障方面可能比使用种子故障更有效。

然而，重要的是要记住，只有空间故障是在测试和操作过程中检测到的真实故障。因此，观察到的空间和其他程序之间的差异的一个合理原因是，后者的故障比空间的真正故障更难被发现。如果这被认为是最合理的原因，那么对空间故障的测试结果因此更可信，并且应该仅根据该程序的结果得出结论。

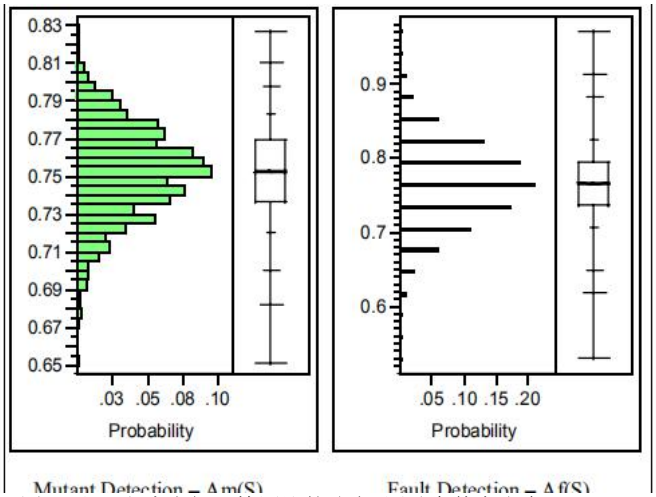

图1。主题程序空间：检测比的分布-测试套件大小为=100

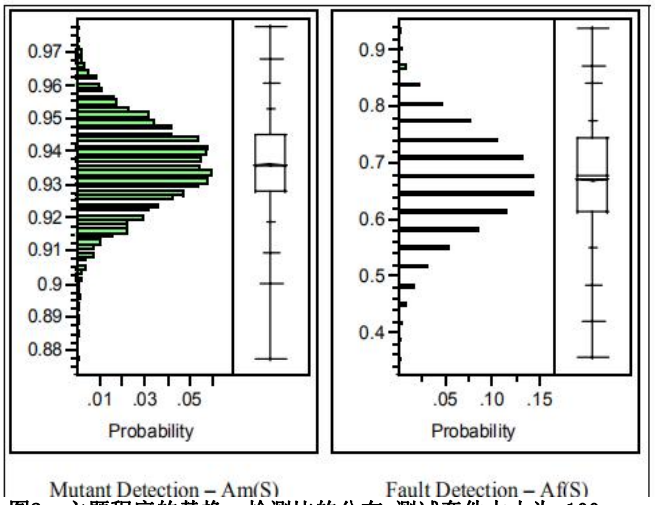

图2。主题程序的替换：检测比的分布-测试套件大小为=100

表3。匹配对t检验结果-测试套件大小=100

	匹配对结果		
主题程序	平均 Af(S)-Am(S)	t比	p值
空间	0.014	16.87	< 0.0001
更换	-0.266	-233.96	0.0000
打印令牌	-0.344	- 158.2	0.0000
印刷机2	-0.061	-59.39	0.0000
进度表	-0.298	- 161.33	0.0000
附表2	-0.327	- 152.19	0.0000
Tca	-0.1128	-57.56	0.0000
Totinfo	-0.1037	- 145.78	0.0000

.24其他解释

除了真实故障和种子故障的相对可检测性外，其他几个因素还可以解释前一节中提供的数据：测试池、程序的内部特征(e。g.，大小)，测试大小套件与测试池和程序的大小的比较，以及突变过程。重要的是要依次考虑它们，以评估我们的第一个解释是否确实是最合理的。

测试池可以包含平均具有不同检测能力的测试用例。然而，关于这些程序[13]的文献表明，这些池是按照相似的过程和相同的结构标准形成的：所有节点、所有边、所有定义-使用对。此外，为了解释我们的结果，这意味着平均测试用例检测能力取决于突变体或故障，与西门子程序相比，该程序的故障检测率为中间材料，突变体的检测率较低。

另一种可能性是，与测试池的大小或程序的大小相比，Am和Af之间的差异随着测试套件大小的增加而增大。（空间比其他程序大得多，测试池也更大。）如果是这样的话，那么随着测试套件规模的减少，我们应该会看到西门子程序的Am-Af更小。然而，对于10、20和50的较小测试套件，结果是相似的：平均(Am-Af)保持相似，对于我们使用的较小测试套件的其他程序，甚至倾向于增加。附录中的图表比较了四种测试套件尺寸的Am和Af；其他西门子程序的图表也很相似。

突变过程可能在某种程度上有偏差，因此突变体在太空上比在西门子程序上更难被杀死。然而，我们遵循了与Space完全相同的突变过程，并且没有明确的原因来解释为什么结果会有所不同。因此，空间比西门子程序大得多这一事实更可能解释较低的突变检测率的原因，因为已知较大的程序更难测试。然而，由于我们拥有的少量主题程序，这种解释既不能证实，也不能证伪，因为所有的西门子程序都很小，规模相似，而且我们没有在空间的连续范围内的观测。

空间故障的检测能力高于西门子程序是我们上面已经讨论过的因素，也是最后要考虑的因素。为了进一步研究这是否是最合理的解释，我们接下来将关注故障和突变体的可检测性，即它们被从跨程序的测试池中随机抽取的测试用例检测到的可能性。

假设西门子手工种子故障比实际空间故障更难检测——而且这不是由于不同测试用例检测能力的变化——我们应该观察以下趋势：（1）空间故障对于任何单个测试用例都比西门子故障更容易检测，（2）空间突变体不应该对于任何单个测试用例比西门子程序突变体更容易杀死。我们将在接下来的两个部分中依次探讨这些预测。

.34、故障可检测性的比较跨程序

图3显示了每个主题程序的比率E(F)的分布，即检测每个故障F的难度（第3.4节）。y轴是每个故障的计算比率，而x轴只是主题程序。x轴上每个受试者程序的水平范围与其相对故障样本大小成正比。例如，我们可以看到，空间和替换的样本量更大，而计划和计划2的样本量相当小（表1）。这些不平等的样本量将需要在我们的分析中考虑到。此外，图中还显示了每个程序的观测结果、总体平均值（跨程序的水平线）、每个程序的特定平均值（跨每个钻石的线）以及它们的平均值的95%置信区间（每个钻石的垂直跨度）。

图3清楚地显示，正如我们所预期的那样，空间比率往往高于其他程序。单因素方差分析(anova)[7,22]显示，总体上，在程序均值之间有统计学上的显著差异(f检验，p值<0.0001)。西门子程序和空间故障的E(F)平均值分别为0.035和0。分别为14个，因此清楚地显示出显著性差异。

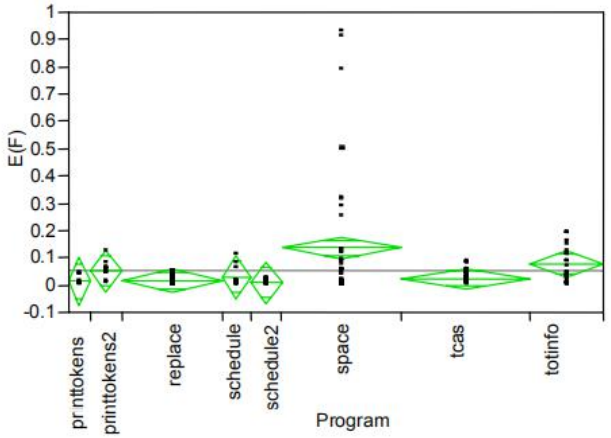

图3。故障检测难度分布-E(F)

进一步的统计测试比较每个西门子程序与空间(使用Bonferroni程序[4]对于使用t检验的多重比较，[22])显示空间仅与Tcas和替换有显著不同（在α=0.05），尽管空间和所有其他程序之间在图形上是可见的差异[5].其他西门子程序缺乏统计意义的一个可能原因是，我们处理小样本，故障数量从7个（打印标记）到23个(Totinfo)。这也可能是由于邦费罗尼手术往往是保守的[7]。这增加了第二类错误的概率，并使其有可能无法检测到合法的重要结果。换句话说，它保证了第一类错误率最多为α（在我们的例子中是0.05），但在现实中可能要少得多。

.44.比较突变体的可检测性跨程序

图4显示了E(M)的分布，即测试案例的比例，杀死每个突变体，为每个受试者的程序（第3.4节）。它的结构与图3相同，除了它看到的是突变体而不是错误。

对于前一节中的错误，方差分析显示，不同程序之间的差异总体上具有统计学意义(Ftest，p值<0.0001)。

当再次使用Bonferroni程序和t检验更仔细地观察每个程序对时，我们观察到空间的平均值明显低于除Tcas外的所有程序（α=0.05），确认了图4中可见的内容。因此，我们可以得出结论，不仅突变体不容易被杀死，而且个别测试案例显示，在空间中检测突变体，在较小程度上检测Tcas。对于空间，如上文所讨论的，这可能是由于这个程序的规模要大得多（见表1）。

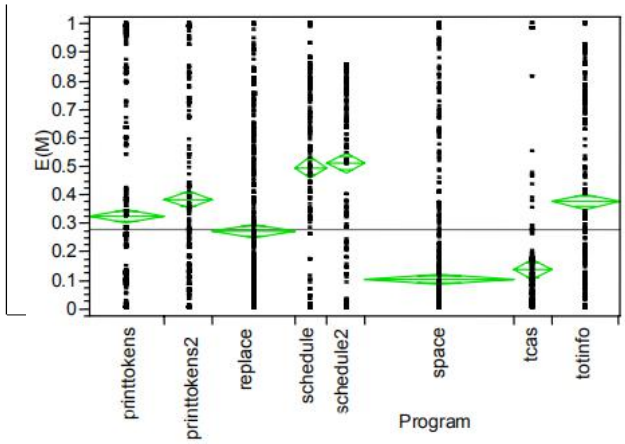

图4。突变体检测难度-E(M)的分布

[4] 这需要使用α划分的比率作为显著性阈值

根据平均比较的数量，在我们的例子中是28个。一个等价的，

[5] ~~非参数的曼-惠特尼检验[7,22]得到相~~同的结果。其他用于多重均值比较的程序，如Tukey的方法，不能被使用，因为我们在[22]程序之间的样本量不相等。

对于突变体和错误的这些反向趋势支持了我们的解释，即不同程序之间的差异不是由于不同的测试用例检测能力，因为这将反而显示出突变体和错误的一致趋势，i。e.，空间测试用例检测到更多的突变体和故障。图3和图4以及他们相关的统计测试结果因此支持我们的猜想，最合理的解释在4.1节的趋势是，故障空间往往更容易检测，因为他们是真正的错误，而不是由人类设计的错误是西门子的程序。

4.5讨论

基于上面的结果我们可以推断，测试套件突变检测比率与故障检测比率在空间，而他们是非常不同的其他程序，是由于故障播种在其他程序可能不代表的容易检测。毕竟，它们并不是真正的故障，人类很可能很难测量种子故障的可探测性。

在首次报道这些程序的论文中，[14]实际上证实了这一结论。哈钦斯等人。明确地指出，从开发人员建议的最初一组大故障中，350个或更多测试用例（从原始测试池）检测到的故障被丢弃[6].这支持了我们上面的猜想，即空间的结果应该被给予更可信度。如果是这样的话，那么我们可以得出结论，基于这里给出的突变算符，突变体确实提供了代表真实错误的测试有效性结果。

它也意味着任何研究人类种子的错误应该仔细解释，因为我们看到通过使用这里使用的示例程序——已广泛使用于以前的研究[8]-one将获得保守的结果在评估故障检测率测试技术。换句话说，这将导致低估测试技术的有效性。这在衡量不同测试技术的成本效益的情况下尤其重要。如果技术A检测到比技术B更多的难以检测到的西门子故障，但a比技术B更贵，我们可能实际上还没有证明a的相对成本效益。

在一些实验设置中，我们可能希望集中于难以检测的故障，因为这些是程序员最麻烦的故障。在这方面，与我们在这里使用的全部突变体集相比，在西门子套件中播种的错误更可取。然而，我们认为，研究人员应该有可能选择困难的突变体，就像选择困难的西门子故障一样，i。e.，通过排除那些被大量测试案例杀死的突变体。如果研究人员这样做了，那么他们就有可能描述整个突变选择过程，从突变操作符到容易排除突变，以一种精确的方式，允许其他研究人员复制。

[6] 他们报告说，大约38%的故障因此被丢弃，18%的故障由于难以检测而被丢弃。

5.结论

这篇论文的主要贡献是对一个基本假设的深入调查，这是迄今为止许多实验软件测试研究的基础：手工产生的故障(e。g.，由经验丰富的工程师)或从突变操作员是代表真实的故障。事实上，大多数实验结果都依赖于种子突变体[1,2,15,16,24]——使用突变操作符——或由经验丰富的开发人员[10,14]选择的错误。此外，这个问题在未来的研究中不太可能消失，因为真正的错误通常太少，无法进行统计分析的实验。因此，即使一个特定的主题程序有真正的错误，人们也经常被迫产生额外的错误。

因此，一个基本的问题是，突变体或人工播种的故障是否可能产生结果，例如在检测能力方面，这代表了人们在真实故障上将获得的结果。我们的分析表明，如果使用仔细选择的突变操作符和删除等效的突变体，突变体可以很好地表明测试套件的故障检测能力。此外，它还显示了使用人类选择的故障的危险，因为在本文研究的系统中，它会导致低估了测试套件的故障检测能力。人类选择的错误的另一个问题更多的是与随着时间的推移建立一个关于测试技术效率的实验结果体的必要性有关。为了做到这一点，研究必须由研究人员重复。当故障是基于一个主观的、未定义的过程来选择的时，这是很难实现的。

我们还应该注意的是，尽管我们在本文中的重点是表明，我们生成的突变体并不比真正的错误更容易检测，但我们的数据也表明，在任何实际程度上，它们并不比真正的错误更难检测。这一事实支持了作为突变测试理论基础的“称职的程序员假设”；也就是说，假设程序员所犯的错误将被杀死突变体的测试套件检测到。

未来的研究当然需要复制我们在本文中报道的研究。我们这样做的方式，应该有助于精确地复制我们的分析，从而将未来的结果与我们的结果进行比较。在面向对象的系统中进行类似的研究也很重要，因为这些系统代表了工业系统中越来越大的份额。最后，由于本文的结果报告随机选择的测试套件相同的大小（尽管测试池构建满足结构覆盖标准），也将是重要的研究结果是否相同的测试套件选择根据各种标准，如代码覆盖标准或操作配置文件标准。

6.确认

非常感谢格雷格·罗瑟梅尔和洪苏克提供的宝贵讨论和建议，并向我们发送了西门子和太空项目以及所有相关的文物。也要感谢所有多年来从事和改进这些主题项目和文物的研究人员。我们还要感谢MikeSowka审阅了这篇论文的草稿。这项工作部分得到了加拿大研究主席(CRC)的资助。杰米的昵称

安德鲁斯、莱昂内尔·布里安德和伊文·拉比奇得到了NSERC业务拨款的进一步支持。

7.参考文献

[1] J.H.安德鲁斯和Y。张，“一般测试结果
检查与日志文件分析，“IEEE在软件工程上的事务，vol。
29 (7), pp.634-648, 2003.

[2] L.BriandY。标签和Y。王，“使用模拟
实证调查测试覆盖标准。*IEEE/ACM国际软件会议*
*工程，爱丁堡，页。*2004年5月86-95日。

[3] T.A.Budd和D。“正确性的两个概念”
*以及它们与测试的关系。*18
(1), pp.31-45, 1982.

[4] D.T.坎贝尔和J。C.*斯坦利，实验和准*
为研究的实验设计，霍顿·米夫林
公司，1990年。

[5] W.陈，R。H.Untch, G.罗瑟梅尔，S。埃尔鲍姆和J。von
“故障暴露-潜在估计能提高测试套件的故障检测能力吗？
*”，“软件测试、验证和可靠性”，第1卷。*12 (4),
pp.197-218, 2002.

[6] R.A.德米洛，R。J.利普顿和F。G.赛沃德，“提示
测试数据的选择：帮助实践程序员，“IEEE计算机，卷。11
(4), pp.34-41, 1978.

[7] J.L.*工程设计，概率和统计学*
*《科学》，达克斯伯里出版社，第5期*th1999年版。

[8] H.做，G。罗瑟梅尔和S。“基础设施支持”
对于软件测试和回归测试技术的控制实验，“美国俄勒
冈州立大学，技术报告04-06-01,1月，
2004.

[9] P.G.法兰克和O。“进一步的实证研究”
测试有效性，”Proc。*6thACM签署了软件基础国际*
研讨会
*工程部，奥兰多（佛罗里达州，美国），pp。*11月153162
1-5, 1998.

[10] P.G.法兰克和S。N.韦斯，“一个实验性的比较”
所有用途和所有边缘充分性标准的有效性。*4th关于测试*
、分析和
*验证，纽约，页。*154164, 1991.

[11] T.L.坟墓，M。J.Harrold, JM..-金，A。波特和G。
“回归测试选择技术的实证研究”，《软件工程与方
*法学报，卷。*10 (2), pp.184-208,
2001.

[12] R.G.哈姆雷特，“用a
*编译器，“IEEE软件工程事务报”，第1卷。*3 (4),
pp.279-290, 1977.

[13] M.更难，J。梅伦和M。D.恩斯特，“改进测试
套件通过操作抽象，“Proc”。*25th*
国际软件工程会议，
波特兰，或，美国，页。2003年5月60-71日。

[14] M.哈钦斯，H。霜冻，T。戈拉迪亚和T。奥斯特兰德
“数据流的有效性实验
基于控制流的测试充分性标准，”序言。*16th*
IEEE软件工程国际会议，
索伦托（意大利），页。191-200年，1994年5月16-21日。

[15] S.金，J。A.克拉克和J。A.麦克德米德，“调查
面向对象测试策略的有效性
突变方法，“软件测试，验证和”
*可靠性，卷。*11 (3), pp.207-225, 2001.

[16] A.M.梅蒙，我。班纳吉和A。Nagarajan，什么测试
我应该用它来进行有效的GUI测试吗？”*IEEE自动化软件*
工程国际会议(ASE‘03)，蒙特利尔，加拿大魁北克，pp
。164-
2003年10月173日。

[17] A.J.“软件测试的调查”
耦合效应，“ACM软件交易”
*《工程与方法论》，第1卷。*1 (1), pp.318, 1992.

[18] A.J.奥福特，A。李，G。罗瑟梅尔，R。H.打开和C。
Zapf，“充分突变算子的实验确定”，《ACM软件工程与
方法学报》，第1卷。5 (2), pp.99118, 1996.

[19] A.J.奥福特和J。“检测等效的突变体和
*可行路径问题，“软件测试，验证和可靠性，卷。*7 (3),
pp.165192, 1997.

[20] A.J.奥福特和R。H.“突变2000：联合起来的
正交的。*突变，圣何塞，加州，美国，pp。*45-
2000年10月55日。

[21] T.J.奥斯特兰德和M。J.“类别分区”
指定和生成功能测试的方法，”
*ACM的通信，vol。*31 (6), pp.676-686,
1988.

[22] J.*大米，数理统计和数据分析，*
杜克斯伯里出版社，2nd1995年版。

[23] G.罗瑟梅尔和M。J.哈罗德，“一个的实证研究”
“安全回归测试选择技术”。*关于软件工程，卷。*24 (6),
pp.401-419, 1998.

[24] P.Thevenod-Fosse，H。韦塞林克和Y。克劳泽特
软件结构测试的实验研究：
确定性与随机输入生成，“Proc”。*21st*
容错计算国际研讨会，
蒙特利尔，加拿大，pp。1991年6月410-417日。

[25] F.I.沃科洛斯和P。G.“实证评价”
文本差分回归测试技术。
IEEE软件维护国际会议，
贝塞斯达，医学博士，美国，页。1998年3月44-53日。

[26] C.沃林，P。Runeson, M.主机，M。C.奥尔森，B。雷格内尔
和A。*魏斯伦，软件工程实验-简介，吉隆坡，2000年。*

8. 附录

图5和图6显示了Am和Af的平均值如何随测试套件大小的函数而变化。我们可以观察到，无论大小如何，Af和Am在空间上都非常相似。然而，对于“替换”，对于其余的程序，差异是很大的，并且往往会随着测试套件大小的减小而增长。

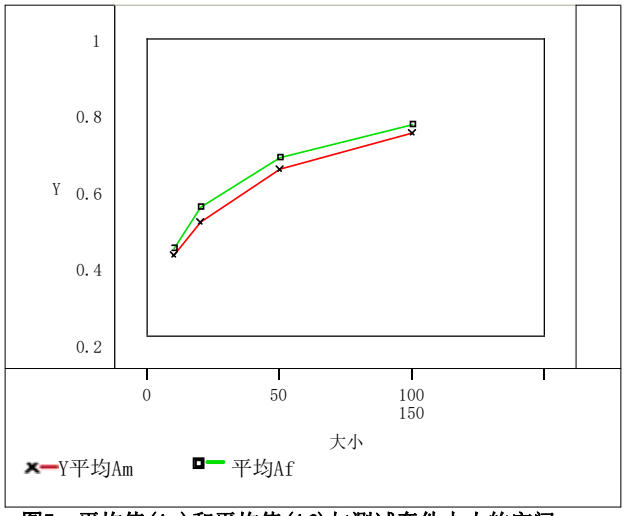

图5。平均值(Am)和平均值(Af)与测试套件大小的空间

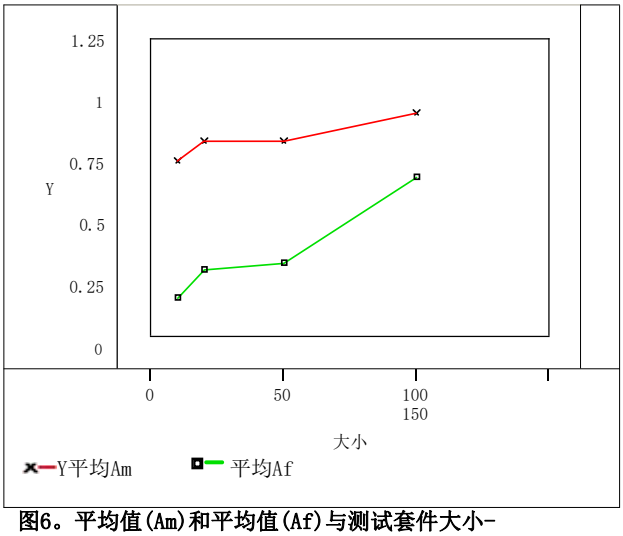

图6。平均值(Am)和平均值(Af)与测试套件大小-更换