# 真实世界测试环境中缺陷预测模型可行性的概率界

UmamaheswaraSharmaB[a],*,1, Ravichandra Sadam[a],2

[a]瓦朗加尔国立理工学院计算机科学与工程系

文章信息

关键词：软件缺陷预测
软件可靠性 机器学习 真实世界测试 Chernoff界

摘要

开发软件缺陷预测（SDP）模型的研究旨在减轻测试人员的工作负担，从而缩短在目标模块上花费的时间。然而，尽管已有大量研究致力于开发预测模型，或尝试缓解开发预测模型时遇到的相关问题，但我们仍不清楚所开发的预测模型在真实世界测试环境中是否真正有效。本文填补了这一研究空白，探究所开发的二元缺陷预测模型在真实世界测试环境中的可行性。由于机器学习（ML）的应用涵盖众多领域，我们希望本文能够为研究实时场景中相关应用所开发预测模型的可行性提供充分依据。

arXiv:2301.06303v1 [cs.SE] 16 Jan 2023

## 1. 引言

软件缺陷预测（SDP）是一个活跃的研究领域，多年来吸引了众多研究人员的关注[1–12]。构建软件缺陷预测模型，旨在帮助软件测试人员更轻松地处理问题代码，而不必在新开发的软件项目中搜寻有缺陷的代码。这些模型基于机器学习（ML）技术构建，并使用从已开发项目中收集的缺陷数据。通常，缺陷数据是针对先前开发的软件项目中[3] 的模块收集的。软件模块的缺陷数据通过广为人知的度量指标套件来表示，例如规模度量指标（如代码行数（LoC））、复杂度度量指标 [14],、Halstead体积度量指标 [15], 等。这些度量指标被视为独立特征，用于训练机器学习模型。之后，利用训练好的SDP模型，确定新开发软件模块的缺陷倾向性。

随着研究的进展，人们提出了几种类型的软件缺陷预测（SDP），包括项目内缺陷预测（WPDP） [3], 跨项目缺陷预测（CPDP）[12, 13] 和异构缺陷预测（HDP） [16]。WPDP模型针对单个软件构建，使用该软件先前开发版本的数据训练模型，并将同一软件新开发版本中的模块信息作为测试数据 [3]。CPDP模型基于从多个源项目收集的缺陷数据构建。此时，当前开发的项目（也称为目标项目）中各模块的类别标签通过使用训练好的CPDP模型来推断。训练CPDP模型的唯一标准是所有源项目和目标项目之间存在共同的度量空间 [12, 17]。而当源项目和目标项目之间不存在共同的度量空间时，HDP模型则可提供解决方案 [16]。然而，各类SDP模型的共同任务都是确定新开发的软件模块的缺陷倾向性。

一旦开发出更好的预测模型，任何组织都可以利用该模型的服务。通常，开发出的预测模型可能会对目标项目产生错误分类预测，因为任何最先进的预测模型在将未见过的样本分类到相应类别时，都会存在最小程度的误差 [18, 19]。此处的错误分类包括假阳性、假阴性，或两者兼有。以下几个示例说明了测试人员面对错误分类预测时可能采取的行动。假设新开发的模块无缺陷，而预测模型将其分类为有缺陷。此时，测试人员会看到假阳性结果，并检查该模块是否正确。相反，假设开发的模块有缺陷，而预测模型将其分类为无缺陷，那么测试人员看到的就是假阴性结果。在这种情况下，测试人员会漏掉这个缺陷模块。此类模块中未被发现的缺陷也称为软缺陷或潜伏缺陷，通常会在运行阶段通过触发此类缺陷模块而被发现；否则，它们将继续潜伏在系统中 [20]。这些潜伏缺陷会危害软件，因为一旦被激活，便可能导致整个系统失效 [20]。

在人工测试流程中，软件会因设计缺陷或恶意逻辑等原因发生故障。 [20,21]。然而，在具有完善测试流程的常规软件开发生命周期（SDLC）中，测试人员会评估已开发的模块，以移除或修改其中的有缺陷内容 [21]。经过多次测试迭代，过

*通讯作者uma.phd@student.nitw.ac.in (U.S. B);ravic@nitw.ac.in (R. Sadam) ORCID(s):[1]印度特伦甘纳邦瓦朗加尔国立理工学院计算机科学与工程系研究学者，邮编 506001[2]印度特伦甘纳邦瓦朗加尔国立理工学院计算机科学与工程系副教授，邮编 506001[3]软件缺陷预测领域的文献将任何软件程序、函数、类或方法都视为一个模块 [3, 6, 8, 12, 13]。

**表 1**所用符号及其对应说明

| 符号 | 说明 |
|---|---|
| $P$ | 一个软件项目 |
| $p$ | 一个概率值 |
| $M_i$ | 项目 $P$ 中的一个软件模块 |
| $S_n$ | 项目 $P$ 中新开发的软件模块集合 $n$ |
| $l$ | 预测为无缺陷的新开发模块数量 |
| $FOR$ | 假遗漏率 |
| $\mathbf{Pr}[-]$ | 离散变量上的概率分布 |
| $X_i$ | 用于衡量危险值的指示随机变量 |
| $X$ | 由 $l$ 独立随机变量之和组成的随机变量， $X_i$， $\|i\| = l$. |
| $\hat{z}$ | 从随机变量中取得的危险值 ，$X$。或 $\hat{z} = X$ |
| $\mathbb{E}[*]$ | 一个期望值 |
| $\mu_{\hat{z}}$ | 随机变量 $X$ 的期望，用于表示预期风险率 |
| $t$ | 时间单位 |
| $z(t)$ | 系统的一般风险率 |
| $\Delta t$ | 时间单位的变化 |
| $F(t)$ | 累积失效分布函数 |
| $R(t)$ | 可靠度函数 |
| $f(t)$ | 失效密度函数或概率分布函数 |
| $T$ | 用于衡量失效概率的随机变量 |
| $\lambda$ | 人工测试的软件任意最佳常数危险率值 |
| $K, \hat{K}$ | 威布尔分布的尺度参数 |
| $m, \hat{m}$ | 威布尔分布中的指数参数值 |
| $\delta$ | 带宽或从期望值测量的区间 |
| $e$ | 一个指数函数 |
| $\hat{R}(t)$ | 根据 $\hat{z}$ 计算得出的基于 SDP 的软件的可靠性 |
| $\mu_{\hat{R}}$ | 根据 $\hat{z}$ 计算得出的基于 SDP 的软件的期望可靠性 |
| $Y_i$ | 用于衡量危险值的指示随机变量 |
| $Y$ | 由 $l$ 个独立随机变量之和构成的随机变量， $X_i$， $\|i\| = l$. |
| $\hat{z}_Y$ | 从随机变量中取出的危险值 ，$Y$。或者 $\hat{z}_Y = Y$ |
| $\mu_{\hat{z}_Y}$ | 随机变量的期望 $Y$ |
| $\hat{R}_Y(t)$ | 根据 $\hat{z}_Y$ 计算的基于 SDP 的软件的可靠性 |
| $\mu_{\hat{R}_Y}$ | 根据 $\hat{z}_Y$ 计算的基于 SDP 的软件的期望可靠性 |

一段时间后，在交付产品之前，系统质量将得到提升 [20, 21]。但是，在使用SDP模型的情况下，测试人员仅依赖预测结果；如果SDP输出任何假负例，那么如上文所述，有缺陷的模块将潜伏在系统中（假设集成测试、系统测试或验收测试未发现这些缺陷）。在这种情况下，很难实现软件系统质量目标。

为了理解潜伏缺陷对已开发的软件系统所产生的影响程度，本文基于概率尾界进行了理论分析。换言之，本文通过概率界，深入探讨已开发的二元缺陷预测模型在真实世界测试环境中的可行性。本文给出的界在很大程度上依赖于一些假设（有关这些假设的详细讨论见第2节和第6节）。初步假设是，已开发模块的实际缺陷倾向未知。随后，我们将每个预测为无缺陷的模块的结果建模为一个相互独立的随机变量。在开发预测模型时，我们假定训练数据、测试数据和总体数据具有相似的分布。

因此，我们为每个误分类缺陷模块（或随机变量）赋予相同的概率值。这里，我们将假遗漏率（$FOR$）（在测试数据集中观测到）作为由于缺陷模块被误分类而导致软件发生单次失效的概率。由于每个随机变量都表示已开发软件中的一个失效条件，随机变量之和（将针对预测为无缺陷的模块定义）就表示该软件在任意时刻的总失效数， $t > 0$。我们将软件中由此产生的故障实例视为“危害实例”。现在，我们考察同一软件的风险率；该软件已由一组测试人员进行了全面测试。

利用 Chernoff界技术 [22, 23], ，我们给出了界，说明使用软件缺陷预测的软件中危害数量少于由人工测试人员测试的同一软件中的危害总数的发生情况（在任意时刻，$t > 0$）。同样，利用相同的技术，从各种风险模型的可靠性角度，我们给出了界，说明使用软件缺陷预测的软件具有比由人工测试人员测试的同一软件（在任意时刻，$t > 0$）更高可靠性的发生情况。

由人工测试人员测试。我们在推导各种证明时，假设人工测试的软件服从风险率的威布尔分布（及其其他补充形式）。例如，假设人工测试的软件服从风险形式的威布尔分布（$Kt^m$）。然后我们推导出一个 $e$的最大上界 $\frac{-(lp-Kt^m)^2}{2lp}$ 作为发生以下情况的概率 - 软件（使用软件缺陷预测）中危害数量较少的情 n 况，相较于同一软件经人工测试后的总危害数量。同样，就采用风险率威布尔分布建模的软件可靠性而言， - 基于风险率的威布尔分布建模，我们的研究得出的是 unerquicklich - ¯推导出 $e$的最大界 $\left[e^{lp(e^{-t}-1)}-\frac{Kt^m}{m+1}\right]^2$ ——— $12e^{lp(e^{-t}-1)}$ 表示使用 SDP 的软件具有高于相同但经人工测试的软件的可靠性的概率。

此外，假设（针对误分类的有缺陷模块定义的）每个随机变量都会生成一个威布尔分布的危害实例，而非系统中的单次失效，我们还推导出了更多界（以风险率和可靠性表示）。

当我们观察到随机变量（假设其表示使用 SDP 模型测试的软件中的危害）与相同但经人工测试的软件的实际风险率之间存在更大偏差时，所推导的界会变得更加精确（甚至趋近于零）。在这种情况下，我们得出结论：所开发的预测模型无效。

### 1.1. 贡献

本研究旨在展示 SDP 模型在真实世界测试环境中的可行性。据我们所知，本研究为软件缺陷预测领域作出了以下独特且重要的贡献：

1. 我们给出了在真实世界测试环境中已开发的缺陷预测模型可行性的紧下界。这些界从软件的风险率和可靠性两个方面进行了讨论。

据我们所知，这是首项在真实世界测试环境中对已开发的二元预测模型进行批判性分析的工作（在本研究中，我们使用软件缺陷预测模型作为分析工具）。

论文结构：第 2 节提出了一些假设，以推导出界。此外，我们还确定了软件失效的概率（由假阴性注入导致）及其分布。第 3 节介绍了风险率、可靠性等可靠性指标。此外，本节还描述了修订后的浴盆曲线，以展示软件可靠性。第 4 节给出了使用切尔诺夫界的证明。第 5 节讨论了威布尔分布的补充内容对可靠性指标的证明。第 6 节详细分析了推导出的证明。第 7 节总结了本研究，并提出了未来可能的研究方向。

## 2. 预备知识

在第1节中，我们讨论了多种类型的SDP，例如跨版本软件缺陷预测、跨项目软件缺陷预测和异构缺陷预测。然而，这些模型旨在对新开发的软件模块进行二元分类（判定其有缺陷还是无缺陷）。我们的目标不受SDP模型类型的影响。也就是说，第4、5和6节中给出的所有证明都适用于任何类型的缺陷预测模型二元分类，因为它们的共同目标都是找出新开发的软件模块是否容易出现缺陷。

现在，定义概率是为软件缺陷预测模型的可行性提供最紧界的第一步。在第2.1节中，我们将详细说明如何根据预测（尤其是利用混淆矩阵）确定概率。在第2.1节奠定基础后，我们将在第2.2节中定义由大量伯努利试验之和推导出的二项分布。第2.3 节将介绍软件缺陷预测模型的实时适用性。

### 2.1. 推导概率

为了给出使用软件缺陷预测模型的软件相较于人工测试软件时，其最大可能可靠性和最小可能危害的紧界，我们首先根据软件缺陷预测模型的预测结果，统计软件系统中发生故障的可能性。其中，最可能导致故障的情况是将有缺陷模块预测为无缺陷。如果预测模型为有缺陷模块赋予无缺陷标签（假设新开发的软件模块中包含缺陷），测试人员就可能漏掉该模块。因此，软件可能会发生故障。

在对软件中的故障进行建模之前，本节将推导一个概率值，即被错误分类的缺陷模块占观测到的无缺陷样本总数的百分比。在定义该概率及其其他相关推导量之前，我们作出以下假设：

现在，以下假设规定了每个新开发软件模块的二元结果：

**假设1.** 软件缺陷预测模型采用二元分类；因此，对软件模块的预测结果要么是有缺陷，要么是无缺陷（但不能同时属于两者）。

也就是说，无论缺陷模块中有多少个缺陷，预测模型都会为该软件模块分配一个二元值。基于这一假设，我们推断，即使软件模块包含多个缺陷，单一原因也可能导致该软件模块发生故障。

**假设2**。 在真实世界测试环境中，某组织采用基于任意批量学习方法构建的传统软件缺陷预测模型。

**表 2**混淆表。

| | | 预测条件 | |
|---|---|---|---|
| | **样本总数 (有缺陷 +无缺陷)** | **有缺陷** | **无缺陷** |
| **实际条件** | **有缺陷** | 真正例（命中） | 假负例（第二类错误或漏报） |
| | **无缺陷** | 假正例（第一类错误，误报） | 真阴性（正确拒绝） |

一般而言，预测模型基于固定的训练数据集或可变的训练数据集（动态数据）构建。如果预测模型使用固定的训练数据集进行训练（在这种情况下，训练数据取自项目的历史数据），我们就称这一训练过程为 批量学习。相反，如果预测模型利用动态的训练数据集来训练模型，我们就称这一学习过程为 在线学习。一般而言，在文献中，软件缺陷预测模型在训练时会使用已发布软件项目的历史数据[3, 13, 16]。

**假设3**。 在开发预测模型时，训练集、测试集和总体的数据分布相似。

这一假设可确保预测模型在所有数据集上都能产生近似的结果。在不失一般性的前提下，所开发预测模型的统计有效性基于这一假设得以保证。也就是说，通常情况下，SDP（WPDP、跨项目软件缺陷预测或HDP）模型是在源数据分布与目标项目数据相似的基础上运行的。因此，这些分布反映了与总体数据相似的分布[11, 12, 16]。

一般而言，我们从混淆表 2 中观察到，预测为无缺陷的模块要么是假阴性，要么是真阴性。软件唯一可能发生失效的情况，是将有缺陷的模块预测为无缺陷模块。下面的假设确保每个误分类缺陷模块只发生一次失效。

**假设 4**。 每个有缺陷模块的误分类都可能导致软件系统发生一次故障。

对于预测为无缺陷的模块，根据假设 1，上述假设确保每个被误分类的有缺陷实例（我们称之为“休眠缺陷”）都可能导致软件系统发生一次故障。通常，在某些情况下，单个缺陷可能导致多次故障，并假设这些故障遵循泊松过程 [24]。但为简化起见，我们假设每个缺陷在软件中都可能导致一次故障。因此，软件中休眠缺陷（或隐性故障）的总数等于假负例实例的总数。

现在，为了衡量预测为无缺陷的模块（即测试数据集）中被错误分类为有缺陷的实例所占的百分比（下文将用它来衡量新开发软件中的故障实例所占的百分比），我们使用一种名为假遗漏率（*FOR*）的评估指标。假遗漏率用于衡量观测到的无缺陷模块中第二类错误所占的百分比。根据混淆表2，假遗漏率定义为假负例数与阴性判定总数之比。如下所示：

$$FOR = \frac{\text{False Negatives}}{\text{False Negatives+True Negatives}} = \frac{\text{False Negatives}}{\text{Negative Calls}} \tag{1}$$

公式 1 保证了目标软件项目中故障实例的百分比[4]。

现在，我们来定义新开发的有缺陷模块被划分到干净类中的概率。根据假设 2，我们假设某组织采用批量学习模型来预测新开发软件模块的缺陷倾向性（该假设确保假遗漏率取某个任意常数），同时根据假设 3 ，测试集与总体集的分布相似（该假设确保预测模型给出的预测结果近乎均匀）。那么，在实时场景中，任何被预测为无缺陷的模块都将以假遗漏率的概率被错误预测。如下所示：

$$p = FOR = \frac{FN}{FN + TN} \tag{2}$$

公式 2 在实时测试环境中是成立的，因为根据假设 2，我们知道测试人员会使用已开发的预测模型（即，这些预测模型在训练过程中使用历史数据），对于每个新开发的模块，如果其被预测为无缺陷，则每个预测为无缺陷的模块成为原本有缺陷模块的概率 $p$ 相似。具体而言，在软件模块的实时测试中，如果这些模块经过测试并被SDP模型预测为无缺陷，则假设这些预测均为错误预测，且每个预测具有相似的概率， $p$。

现在，利用概率 $p$ （由公式2得出），我们来定义预测为无缺陷的模块集合的分布。

## 2.2. 定义分布

假设1-4 对于定义概率分布至关重要。本节结束时，我们将得出软件系统中的故障事件总数以及预期故障事件数，这些数据均由

[4]通常，SDP模型使用历史源项目数据进行训练（我们使用这些增强数据来训练预测模型），并使用目标项目数据进行验证。

软件缺陷预测模型观测得到。由于我们不知道新开发的软件模块的实际类别标签，因此将每个新开发的软件模块的预测结果视为一个随机变量。在所有随机变量中，我们关注的是用于衡量软件项目中误分类缺陷模块的随机变量。以下情景正式描述了使用软件缺陷预测模型的软件中的故障。

假设 $n$ 项目中开发了新的软件模块， $P$，并使用SDP模型对每个新开发的软件模块进行独立测试，以预测其类别标签。令集合 $S_n = \{M_1, M_2, \ldots, M_n\}$ 表示项目中新开发的 $n$ 软件模块集合，$P$（这些模块的实际类别标签未知）使用 SDP模型预测其类别标签。假设新开发的 $n$软件模块最初要么有缺陷，要么无缺陷（但不能同时属于两者）。请注意，有缺陷的模块可能包含多个缺陷。但为简化起见，根据假设1，缺陷数量将转换为二元值。现在，使用 SDP模型，我们观察集合 $S_n$中模块的预测结果。假设$l$ 个软件模块在 $n$ 个软件模块中被预测为干净类。请注意，被预测为 $l$干净类的模块最初可能有缺陷，也可能无缺陷。现在，为了统计 $l$ 个模块中被错误预测为有缺陷的模块数量，我们定义一个指示随机变量：若新开发的模块 $M_i$，$|i| = l$被错误地分类为干净类，则取值为 1；若新开发的模块 $M_i$ 被正确地分类为干净类，则取值为 0。定义如下：

$$X_i = \begin{cases} 1, & \text{if the module } M_i \text{ is wrongly classified into clean} \\ 0, & \text{if the module } M_i \text{ is correctly classified into clean} \end{cases} \tag{3}$$

在此，我们定义指示随机变量 $X_i,\ i \in \{1, 2, \cdots, l\}$ 用于统计使用缺陷预测模型的软件中的故障实例数量。

根据假设2，由于我们设定某个组织会使用传统软件缺陷预测模型（即通过批量学习开发的模型）的服务，并且根据假设3，我们确保每个新开发模块的预测值具有相似的概率，因此，每个被错误归类的预测为干净的模块都被赋予概率 $p$。其表示如下：

$$\mathbf{Pr}[X_i = 1] = p, \text{ and, } \mathbf{Pr}[X_i = 0] = 1 - p, \text{ for } 1 \le i \le l. \tag{4}$$

其中 $X_i$ 是方程 3中定义的指示随机变量。由于方程 3中定义的每个指示随机变量 $X_i$、$i \in \{1, 2, \cdots, l\}$在取值为 1 时的概率为 $p$（类似地， $X_i$ 取值为 0 的概率为 1-$p$），因此这是一次伯努利试验。请注意，取值 $p = 0$ 表示预测结果中不存在假负例（在测试集上），因此 SDP 模型能够准确地识别有缺陷的模块。相反，取值 $p = 1$ 表示预测结果中存在真负例，因此预测模型无法准确识别实际的干净模块。由此，所有预测为无缺陷的模块都属于假负例类别。现在，以下假设对于给出所有证明（见第 4、5和6节）至关重要：

**假设5**。针对目标项目，软件缺陷预测模型应至少产生一个假负例和一个真负例结果。

该假设确保概率（$p$）的取值位于区间（0，1）内。在第4、5和6节中，我们基于该假设给出了所有证明。不过，支持这一假设的事实是，根据Herbold等人 [17], , 大多数软件缺陷预测模型在目标数据集上的表现不佳。此外，要找到能够在特定应用中始终稳定运行的最佳分类器仍然是一项艰巨的任务 [25]。因此，我们认为这一假设在真实世界测试环境中成立。

现在，利用以下伯努利随机变量组， $X_i,\ i \in \{1, 2, \cdots, l\}$，我们定义二项分布。

### *2.2.1.* 故障事件的二项分布：

由于 SDP模型独立且随机地将每个已开发的软件模块作为输入，以确定其缺陷倾向性，因此每个 $X_i$，$i \in \{1, 2, \cdots, l\}$ 都会成为一个伯努利随机变量。下面的讨论将聚焦于随机变量 $X$，该随机变量由具有相同概率的独立伯努利随机变量之和定义。因此，随机变量 $X$ 服从二项分布。表示如下：

$$X = \sum_{i=1}^{l} X_i \tag{5}$$

方程 5 中推导出的随机变量实际上就是风险率（在部署软件后的任意时刻 $t > 0$，对于基于 SDP 的软件而言）。设 $\hat{z}(t)$ 是使用 SDP模型进行测试的软件的风险率，则有：

$$\hat{z}(t) = X \tag{6}$$

由于假设伯努利随机变量 $X_1$、$X_2$、…、$X_l$ 相互独立，因此如上所述，$l$ 个相同伯努利试验之和服从二项分布。二项分布的均值，或软件中预期的危害实例数，推导如下（利用期望的线性性）：

$$\mu_{\hat{z}} = \mathbb{E}[X] = E\Big[\sum_{i=1}^{l} X_i\Big] = \sum_{i=1}^{l} \mathbb{E}[X_i] = \sum_{i=1}^{l} \big[1 * \mathbf{Pr}[X_i = 1] + 0 * \mathbf{Pr}[X_i = 0]\big] = \sum_{i=1}^{l} p = lp \tag{7}$$

公式7给出了使用软件缺陷预测模型测试的软件中预期失效（危害）事件数，前提是假定每个错误预测为有缺陷的模块都会导致一次失效（根据假设4）。请注意，本文通篇将故障率和风险率这两个术语互换使用。

注意，根据公式6，在实时场景中，软件部署（或发布）给客户后，系统中可能存在 $X$ （或者，平均存在 $\mu_{\hat{z}}$）个缺陷（因此，每个有缺陷实例都会产生一次失效）。现在，我们通过将随机变量 $X$ （该随机变量用于衡量使用SDP模型进行测试的软件中的危害）与由测试人员团队测试的同一软件的风险率进行比较，给出紧界。我们使用称为切尔诺夫界的尾部不等式技术给出证明。二项分布的期望（根据公式 7）对于使用切尔诺夫界给出证明而言是必要且充分的。第 4、5 和 6 节提供了详细的证明列表。

### 2.3. 实时适用性

假设某组织按照一套固定的规格说明开发一款新软件（包含 $n$ 个模块），旨在满足已收集的需求。软件开发完成后（假设开发人员遵循特定的软件开发流程），测试人员团队将对已开发的软件进行测试，以消除系统中存在的缺陷。常见的策略是使用一组测试用例对各个模块进行测试。假设软件中的每个模块都旨在按照编写完善的规格说明运行，而这些规格说明将成为已开发软件模块的测试用例 [20]。在可用时间内（因为任何测试团队都必须在规定时限内完成软件测试），测试人员团队将仔细测试各个模块，以发现（并随后消除）缺陷。经过对软件的全面修订后，大多数会导致软件失效的缺陷都将从系统中消除。然而，显而易见的是，许多软件系统仍会因各种原因而失效，原因包括测试实践不当以及意外的外部因素。

如第1节所述，SDP模型旨在取代人工测试，简化测试工作，并预测已开发软件模块的缺陷倾向性。测试人员根据预测结果，对预测为有缺陷的模块进行代码走查，以移除任何受损代码。但如前所述，如果有缺陷的模块被预测为无缺陷，软件就有可能发生失效。

在上述两种情形下，软件系统都可能发生故障。在确定两种故障数量（一个是在人工测试的软件中观测到的数量，另一个是一个随机变量）之间的偏差之前，必须满足以下假设。

**假设6**。对于人工测试和使用*SDP*模型进行测试，软件是相似的。

这是开展本研究的一项基本且重要的假设。由于我们希望找出可行性，因此我们在下文给出证明：使用 SDP 模型测试的软件的风险率相较于同一软件经人工测试后的风险率存在偏差。此外，我们还利用使用上述 SDP 模型测试的软件的可靠性相较于同一软件经人工测试后的可靠性所产生的偏差来给出证明。

请注意，根据假设6，尽管人工测试和使用SDP模型测试这两种情况下的软件相似，但模块数量和故障事件可能有所不同。这是因为人工测试会采用典型的结构化方法，从新开发的软件中去除死代码、不可达代码或一些冗余代码。因此，在发布产品之前，经过全面测试后，大部分此类代码都会被去除，同时保留原有功能。这可能导致产品发布前的模块总数（与最初开发的模块数量相比）发生变化。同样，使用SDP模型测试软件时，测试人员也会测试预测为有缺陷的模块。对于这类模块，经过全面测试后，大部分不需要的代码都会被去除。因此，同一软件中的模块总数可能会发生变化。因此，我们可能会看到两个软件系统的危害总数存在差异，其中一个由SDP模型测试，另一个则由测试人员团队测试。

现在，以下假设确保了使用 SDP 模型进行测试的软件中存在危害：

**假设7.** 对于被错误预测为有缺陷的模块，集成测试、系统测试或验收测试均未能发现这些缺陷。

这在现实世界的测试环境中虽不常见，但确有可能发生。由于不可能通过人工测试，使用一套穷尽所有情况的测试用例来测试软件，因此软件系统中不可避免地会存在隐藏缺陷。于是，最终用户可能会在触发此类隐藏缺陷后遭遇故障。同样，对于使用 SDP 模型的软件，即使测试已经完成，缺陷仍不可避免 [3, 20]。具体而言，虽然 SDP 模型通过省去单元测试减轻了测试人员的工作量，但测试人员仍可能在集成测试、系统测试或验收测试期间遇到缺陷。然而，根据假设 7，软件发布后，软件系统中仍会存在 $X$ 个危害（或者平均而言， $\mu_{\hat{z}}$ 个危害）。

除了上述假设外，以下假设确保已维护的缺陷模块不会产生危害：

**假设8**。 已修复的缺陷模块不会退化。

基于这一假设，我们只能从预测为无缺陷的模块中推导出危险条件。放宽这一假设，可能会使我们能够预测维护后的代码中发生危险事件的可能性。

现在，以下假设确保了估算两个软件系统（即使用软件缺陷预测进行测试的软件，以及另一个经人工测试的软件）的危害和可靠性时所采用的时间间隔相近。

**假设9**。 这两个软件系统（即使用软件缺陷预测进行测试的软件，以及另一种经人工测试的软件）都将在时间*0*发布，并在时间区间 *[0, t]*内测量其危害和可靠性。

现在，为了验证二项分布适用于实时场景的情况，根据假设 9，所有新开发的模块都在时间0之前使用SDP模型进行测试。也就是说，由于SDP模型会预测新开发的模块是否有缺陷或无缺陷，我们假设测试人员会在发布产品之前处理预测为有缺陷的模块。之后，对于使用SDP模型的软件系统，将在时间区间 $[0, t]$内测量其风险率和可靠性。这便于计算使用SDP模型的软件的可靠性指标。同样，根据假设 9，对于经人工测试的软件，测试人员会在发布产品之前进行完整的系统测试。经过全面的系统测试后，软件将在时间0部署，之后将在时间区间$[0, t]$内测量风险率和可靠性。与使用SDP模型测试软件的情况类似，此假设便于计算经人工测试的软件的可靠性指标。

## 3. 软件的风险率与可靠性

在上一节中，我们假设软件已部署，并估算了使用SDP模型的软件中可能发生的危害数量的期望值。之后，这些危害（一个随机变量）将用于定义软件系统的可靠性。为了根据风险率计算可靠性，我们在第 3.1 节给出了这些可靠性指标的定义及其相互关系。可靠性指标的定义对于人工测试的软件和使用SDP模型测试的软件都适用。

一般而言，硬件系统的可靠性用浴盆曲线表示 [20]。然而，软件可靠性并不具有与硬件相同的特征 [26]。因此，我们在第3.2节讨论软件可靠性修订后的浴盆曲线的详细内容。在第3.3节中，我们将讨论一种广泛使用的风险模型——威布尔分布。此外，我们还将根据风险模型的威布尔分布推导软件的可靠性。

### 3.1. 软件可靠性指标

本节定义并解释风险率和可靠性等可靠性指标之间的关系。我们根据 Lyu 在 [20] 中的研究，改编了可靠性指标的基本定义。

#### *3.1.1.* 风险率

风险率是指在系统存活至时间 $t$ [20, 26] 的条件下，系统在时间 $t$ 发生失效的瞬时速率。其表达式如下：

$$z(t) = \lim_{\Delta t \to 0} \frac{F(t + \Delta t) - F(t)}{\Delta t R(t)} = \frac{f(t)}{R(t)} \tag{8}$$

其中， $f(t)$和 $F(t)$分别为概率密度函数（pdf）（或失效密度函数）和累积分布函数（cdf）。 $R(t)$为可靠度函数，用于衡量在时间$t$时成功的概率。在本文中，我们假设已获得人工测试的软件的风险率。因此，无需使用 $f(t)$和 $F(t)$ 等函数进行估计。

#### *3.1.2.* 可靠性

在定义可靠性之前，我们假设所关注的随机变量是软件的失效时间，即 $T$。现在，软件可靠性被定义为：在指定环境 [20, 26]中，软件在指定时间段内无失效运行的概率。形式化地表示为：

$$R(t) = Pr[T > t] = 1 - F(t) = \int_t^{\infty} f(x)dx \tag{9}$$

#### *3.1.3.* 风险率与可靠性的关系

经过几步代换，并对方程8 和 9进行推导后，可得风险率与可靠性之间的关系，如下所示 [20, 26]:

$$R(t) = e^{-\int_0^t z(x)dx} \tag{10}$$

方程10 有助于推导各种危害模型的可靠性值。对于采用软件缺陷预测模型的软件和人工测试的软件，根据风险率估算可靠性的方法相似。

### 3.2. 软件可靠性的浴盆曲线

一般来说，每当出现缺陷，或认为有助于增强软件功能的情况时，测试人员团队通常都会对软件系统进行维护。因此，随着时间推移，通过移除已发现的缺陷对系统进行修改，软件中的故障事件将逐渐减少至最低水平。由此可见，软件不会随着时间推移而磨损，但可能会因规格不当或被误解、输入数据错误、算法错误、程序逻辑错误等而发生故障。 [20]。在给定时期内，任何软件中可能出现的典型危险率曲线如图1所示。

与硬件相比，软件中的风险率建模有所不同。在 [27], Keene 提供了一份部分清单，列出了软件系统风险率变化的不同原因。在 [26],Mary Hartz 等人提出了一条典型的修订后的浴盆曲线，如图1所示。从图1可以看出，软件的风险率分为

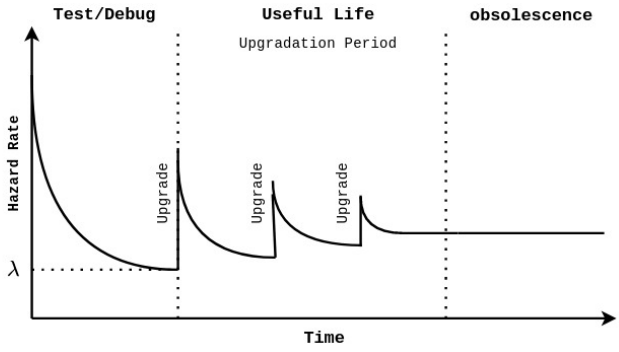


**图1：** 软件可靠性的典型修订后的浴盆曲线 [26]。

三个区域，即测试/调试、升级以及稳定/过时区域。图 1 中的区域 I 表明，部署之后，软件可能会因书面软件程序（模块）中存在的缺陷被触发而经历更多故障。因此，软件在该区域的风险率可能较高。质量控制机制和初始产品测试策略有助于消除系统中已发现的缺陷。在此区域，风险率往往随时间推移而下降。

图1 中的区域II代表软件的升级阶段。在此区域中，软件系统会进行更多升级，这些升级可能是为了移除有缺陷的功能，或增强软件中的功能 [26]。因此，每当软件系统进行新的升级时，风险率都会发生波动。有时，软件系统中的错误修复可能会导致级联缺陷（继而导致级联故障）；有时，软件的每次升级都会使系统中的故障事件数量减少。

图1 中的区域III代表软件的稳定状态，因为软件不会进行任何新的升级。或者，在此区域中，软件通常被视为已经过时，表明系统没有提供功能更新的动机。然而，如果软件仍在使用，最终用户在此时间段内可能会遇到更少的故障事件[20, 26]。因此，与硬件系统中的风险率相比，该区域内系统的风险率被视为任意常数值 [20]。由于软件会随着每次升级而变得更加复杂，系统中可能会消除更多缺陷。因此，软件系统将在更长时间内保持稳定。

由于软件在一段时间内风险率存在变化（见图1），我们在假设风险模型服从威布尔分布的基础上给出了证明。威布尔分布能够为一段时间内的风险率提供不同的函数形式。相关文献展示了对软件的许多复杂风险函数和可靠性模型进行估计的方法 [20,26, 28, 29]。然而，为了便于计算界，我们采用了软件中风险的威布尔分布形式。软件中风险率的威布尔模型见第3.3节。

### 3.3. 威布尔分布

在许多情况下，给定时间段内软件的风险率并不遵循特定的函数形式 [20, 21]。为了近似各种危险率曲线，我们采用一种称为威布尔分布的风险率形式。

**定义1**。 对于任何软件系统，危害函数 ($z(t)$) 的威布尔分布定义为 [20]:

$$z(t) = Kt^m, \text{ for some } K > 0, m > -1, \text{ and time } t > 0 \quad (11)$$

威布尔分布是一个与时间相关的函数，通过适当选择两个参数 $K$ 和 $m$，即可近似各种特定的风险率。例如，在 $m$取固定值时，参数 $K$ 的变化仅会导致风险率的垂直振幅发生变化。对于一个简单情形，在威布尔分布中将 $m$ 设为 0，并将 $m$ 设为 1，可分别得到恒定风险率 和线性递增的风险率。在第 5节中，我们给出了软件中可能出现的各种威布尔风险率分布补充形式的证明。这些补充形式可能使我们能够在构造的证明中采用最相关的风险形式。

现在，根据危害率的威布尔分布（参见定义 1），下面的引理定义了软件的可靠性：

**引理1**。 对于威布尔风险模型 $z(t) = Kt^m$，若某个 $K > 0, m > -1$ 且时间为 $t > 0$，其可靠性为：

$$R(t) = e^{\frac{-Kt^{(m+1)}}{m+1}} \quad (12)$$

证明。 由于我们假设软件中的危害服从威布尔分布，将 $z(t)$ 代入 $R(t)$（即代入公式 10）可得：

$$R(t) = e^{-\int_0^t Kx^m\,dx} \quad (13)$$

现在，化简公式 13后，即可得到所需形式的可靠性。 □

这里，我们用 $z(t)$ 和 $R(t)$ 分别表示人工测试软件的风险率和可靠性。需要注意的是，对于参数$K$ 和 $m$的任意估计值，以及任意时间 $t$，函数 $z(t)$ 和 $R(t)$ 都将取某些任意实数值。

## 4. 证明

在第4.1节中，我们证明了使用软件缺陷预测（SDP）的软件，其危害数量可能少于人工测试的软件。证明中推导出了一个紧下界。同样，在第4.2节中，我们证明了与

人工测试的软件的可靠性相比，使用 SDP 的软件可能具有更高的可靠性。为此，证明中推导出了一个紧上界。为了给出紧上界，我们最终在证明中展示了另一种形式的下界。

### 4.1. 软件风险率方面的紧下界

在第 2.2 节中，我们使用随机变量（即 $\hat{z}(t) = X$）推导了使用 SDP 模型测试的软件中的危害（故障）实例数。现在，以下定理定义了随机变量 $X$ 低于人工测试软件的风险率的偏差，$Kt^m$（实际上，远低于预期风险率 $\mu_{\hat{z}}$）。

**定理 1**。 设 $X_1, X_2, \ldots, X_l$ 为独立伯努利试验，满足对于 $1 \leq i \leq l$, $Pr[X_i = 1] = p$，其中 $0 < p < 1$。那么，对于 $X = \sum_{i=1}^{l} X_i$, $\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} p = lp$，参数 $\exists$ , $K > 0$, $m > -1$，时间 $t > 0$，以及威布尔危害率 $z(t) = Kt^m$:

$$Pr[X < Kt^m] < e^{\frac{-(\mu_{\hat{z}} - Kt^m)^2}{2\mu_{\hat{z}}}} \tag{14}$$

证明。 我们知道，对于某些 $0 < \delta \leq 1$ 和 $\mu$，使用 Chernoff 界，偏离期望 $\mu$ 较远的独立伯努利试验之和 $X$ 的下尾界为 [30]:

$$Pr[X < (1-\delta)\mu] < e^{\frac{-\mu\delta^2}{2}} \tag{15}$$

这里， $(1-\delta)\mu$ 表示偏离期望 $\mu$ 的值，带宽为 $\delta$。

现在，我们希望得到一个紧下界，使随机变量 $X$远低于人工测试的软件的风险率 $Kt^m$。这里，对于某些 $K > 0$, $t > 0$, 以及 $m > -1$，假设值 $Kt^m$ 低于期望 $\mu_{\hat{z}}$，在给定时间段 $[0, t]$内。也就是说， $Kt^m$ 是从期望值 $\mu_{\hat{z}}$向左偏移的任意常数。将 $(1-\delta)\mu$ 的值与 $z(t)$ 在公式 11中的值相等，可得:

$$(1-\delta)\mu = Kt^m$$

$$\Rightarrow \delta = 1 - \frac{Kt^m}{\mu} \tag{16}$$

由于我们要寻找随机变量（$X$）相对于人工测试软件风险率的偏差的紧下界，因此在上式中，我们将$\mu$ 替换为使用软件缺陷预测模型测试的软件的期望风险率 $\mu_{\hat{z}}$。现在，根据方程 7，我们知道使用软件缺陷预测模型的软件中预期的故障数量（假定这些故障是由休眠缺陷模块导致的）:

$$\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} \mathbb{E}[X_i] = \sum_{i=1}^{l} p = lp$$

现在，将 $\delta$（来自方程 16）和 $\mu_{\hat{z}}$（来自方程7）代入方程 15，即可得到随机变量与人工测试软件风险率之间偏差的紧下界。如下所示:

$$Pr[X < Kt^m] < e^{\frac{-\mu_{\hat{z}}\left[1 - \frac{Kt^m}{\mu_{\hat{z}}}\right]^2}{2}} \tag{17}$$

经过几步简化即可完成证明。 □

定理1 表明，在基于 SDP 的软件中，较少危害的发生概率低于由人工测试同一软件时所有危害的发生概率，并且其上界为$e^{\frac{-(\mu_{\hat{z}} - Kt^m)^2}{2\mu_{\hat{z}}}}$最大值。可以看到，指数中的项为负。因此，分子中诸如 $\mu_{\hat{z}}$和 $Kt^m$这样的项之间若存在较大偏差，就会得到更紧的界。这样一来，如果所开发的预测模型与测试人员的质量相当，项 $e^{\frac{-(\mu_{\hat{z}} - Kt^m)^2}{2\mu_{\hat{z}}}}$ 就会趋近于 1。同样，如果所开发的预测模型项 $e^{\frac{-(\mu_{\hat{z}} - Kt^m)^2}{2\mu_{\hat{z}}}}$ 型的表现

el

与测试人员的质量相比很差。

### 4.2. 软件可靠性的紧上界

本节给出一个引理，用于根据采用SDP模型进行测试的软件的风险率计算其可靠性。如上文所述，此处假设采用SDP模型进行测试的软件的风险率是一个随机变量， $X$。

**引理 2.** 设 $X_1, X_2, \ldots, X_l$ 为独立伯努利试验，则对于软件系统中的危害 $\hat{z}(t) = X = \sum_{i=1}^{l} X_i$，其可靠性为:

$$\hat{R}(t) = e^{-Xt} \tag{18}$$

证明。 我们知道，采用SDP模型进行测试的软件中的危害是一个随机变量 $X$。然后利用公式 10:

$$\hat{R}(t) = e^{-\int_0^t X dx} \tag{19}$$

ˆ此处，我们使用 $R(t)$ 表示采用SDP模型进行测试的软件的可靠性。现在，简化公式 19 即可完成证明。 □

现在，为了求出随机变量偏差的紧上界， $e^{-Xt}$，即远离人工测试软件可靠性的偏差，$e^{-Kt^{m+1}}$， $e^{m+1}$ 我们需要计算基于 SDP的软件的期望可靠性。根据引理 2，我们知道使用 ˆSDP模型测试的软件的可靠性。现在，期望可靠性（$\mu_{\hat{R}}$ 或 $\mathbb{E}[R(t)]$）可推导如下:

$$\mathbb{E}[\hat{R}(t)] = \mu_{\hat{R}} = \mathbb{E}[e^{-Xt}] = \mathbb{E}\left[e^{-t\sum_{i=1}^{l} X_i}\right] \tag{20}$$

由于假定随机变量（$X_i$s）相互独立，指数中的各项之和将变为指数项的乘积。表示如下：

$$\mathbb{E}\left[e^{-t\sum_{i=1}^{l}X_i}\right]=\prod_{i=1}^{l}\mathbb{E}\left[e^{-tX_i}\right] \tag{21}$$

这里，随机变量 $e^{-tX_i}$ 以概率 $p$ 取值 $e^{-t}$，以概率 $1-p$取值 1。现在，利用这些值，我们可以从上面的等式得到：

$$\prod_{i=1}^{l}\mathbb{E}\left[e^{-tX_i}\right]=\prod_{i=1}^{l}\left[pe^{-t}+1-p\right]=\prod_{i=1}^{l}\left[p(e^{-t}-1)+1\right] \tag{22}$$

我们知道，$1+x<e^x$。现在，将此不等式与$x=p(e^{-t}-1)$结合，我们重写公式 22 以得到期望可靠性：

$$\mathbb{E}[\hat{R}(t)]=\mu_{\hat{R}}=\prod_{i=1}^{l}\left[p(e^{-t}-1)+1\right]<\prod_{i=1}^{l}e^{p(e^{-t}-1)}=e^{lp(e^{-t}-1)} \tag{23}$$

现在，根据引理 1 和2，以下定理定义了随机变量 $e^{-Xt}$ 相对于其期望 $\mu_{\hat{R}}$的偏差。

**定理 2.** 设 $X_1$、$X_2$、…、$X_l$为独立伯努利试验，满足 $1\le i\le l$、$Pr[X_i=1]=p$，其中$0<p<1$。那么，对于 $X=\sum_{i=1}^{l}X_i$、$\mu_{\hat{R}}<e^{lp(e^{-t}-1)}$、$\exists$，参数 $K>0$、$m>-1$，时间 $t>0$，以及 $Weibull_{-Kt^{m+1}}$可靠度函数，$e^{\ m+1}$：

$$Pr\left[e^{-Xt}>e^{\frac{-Kt^{m+1}}{m+1}}\right]<e^{-\left[e^{lp(e^{-t}-1)}-\frac{Kt^m}{m+1}\right]^2\frac{1}{2e^{lp(e^{-t}-1)}}} \tag{24}$$

证明。该定理的证明与下尾的证明非常相似，正如定理1中所述。与之前一样，

$$\begin{aligned}Pr\left[e^{-Xt}>e^{\frac{-Kt^{m+1}}{m+1}}\right]&=Pr\left[e^{Xt}<e^{\frac{Kt^{m+1}}{m+1}}\right]\\&=Pr\left[Xt<\frac{Kt^{m+1}}{m+1}\right]=Pr\left[X<\frac{Kt^m}{m+1}\right]\end{aligned} \tag{25}$$

现在，我们希望为随机变量 $X$与值 $\frac{Kt^m}{m+1}$相差甚远的情况求得一个紧下界。这里，对于某个$K\succ 0$，$t>0$，且 $m>-1$，假设在给定时间段 $\frac{Kt^m}{m+1}$ 内，一值$\mu_{\hat{R}}$低于期望 $[0,t]$。现在，将式 $(1-\delta)\mu$ 中的15 与方程 $\frac{Kt^m}{m+1}$ 中的25 相等，可得：

$$\begin{aligned}&(1-\delta)\mu=\frac{Kt^m}{m+1}\\&\Rightarrow\delta=1-\frac{Kt^m}{(m+1)\mu}\end{aligned} \tag{26}$$

由于我们要计算随机变量$e^{-Xt}$相对于人工测试的软件的可靠性的偏差紧上界，在上述方程中，我们将 $\mu$ 替换为使用 SDP模型测试的软件的期望可靠性$\mu_{\hat{R}}$ 。将软件的期望可靠性（由方程23得出）代入上述方程，即可得到 $\delta$的最终值。

$$\Rightarrow\delta=1-\frac{Kt^me^{-lp(e^{-t}-1)}}{(m+1)} \tag{27}$$

这里，$\delta$ 表示期望的带宽值，$\mu_{\hat{R}}$。现在将 $\delta$（来自公式27）的值代入公式15，则：

$$\begin{aligned}&Pr\left[e^{-Xt}>e^{\frac{-Kt^{m+1}}{m+1}}\right]=\\&Pr\left[X<\frac{Kt^m}{m+1}\right]<e^{\frac{-e^{lp(e^{-t}-1)}\left[1-\frac{Kt^me^{-lp(e^{-t}-1)}}{(m+1)}\right]^2}{2}}\end{aligned} \tag{28}$$

简化后，证明即告完成。 □

定理2 表明，与由人工测试人员测试的同一软件相比，使用SDP模型进行测试的软件获得更高可靠性的概率受到以下界限约束

其最大值为 $e^{-\left[e^{lp(e^{-t}-1)}-\frac{Kt^m}{m+1}\right]^2\frac{1}{2e^{lp(e^{-t}-1)}}}$。与定理1的结果类似，我们可以观察到，指数项中的项为负。指数分子中的项（例如 $e^{-lp(e^{-t}-1)}$和 $\frac{Kt^m}{m+1}$）之间若存在较大偏差，将使界限更加紧密。这样，当开发的预测模型达到与测试人员相当的质量时，$e^{-\left[e^{lp(e^{-t}-1)}-\frac{Kt^m}{m+1}\right]^2\frac{1}{2e^{lp(e^{-t}-1)}}}$趋近于1。同样，当开发的预测模型与测试人员的质量相比表现较差时，$e^{-\left[e^{lp(e^{-t}-1)}-\frac{Kt^m}{m+1}\right]^2\frac{1}{2e^{lp(e^{-t}-1)}}}$趋近于0。具体而言，如果人工测试的软件的估计可靠性与（随机变量的）期望相差较大，那么界限会更加紧密。

定理1 和2 为研究所开发的软件缺陷预测模型在真实世界测试环境中的可行性提供了理论依据。

利用定理1和2，在第5节中，我们针对威布尔分布的补集给出了更多界（包括风险率和可靠性方面的界）。第5 节中给出的所有证明，均用于理解威布尔分布在不同形式下界的变化情况。

## 5. 补充证明

本节基于威布尔分布的补充材料给出证明。本节中的所有证明均基于定理1 和2。在每个小节中，我们都会给出一种可能的情形，以观察风险率的特定函数形式。本节中的所有证明均用于分析以下模型的可行性：

与可靠性指标的某些特定形式相比，所开发的预测模型（就风险率和可靠性而言）。这有助于纳入并分析各种危害函数（以及可靠性函数），从而得出精确的界。

## 5.1. 非线性递减风险模型

在大多数情况下，软件中的危害通常表示为随时间非线性递减的函数 [20]。例如，从图 1可以观察到，软件的每次升级都会导致软件系统在早期出现故障。因此，在早期阶段，质量控制和初始产品测试团队会剔除不达标的功能，以避免风险率升高。经过数轮测试迭代后，系统中的危害会逐渐降低至最小值。表示非线性递减函数的最简单模型是逆时间函数。也就是说，随着时间推移，系统中的危害会逐渐降低，最终系统中的危害变得更少。需要注意的是，任何复杂函数都可以替代我们用于表示非线性递减风险模型的函数。

在第 5.1.1 节中，我们为使用软件缺陷预测的人工测试软件提供了一个更紧的下界，用于衡量其软件达到最小危害的可能性。类似地，在第 5.1.2 节中，我们为使用软件缺陷预测的软件提供了一个紧上界，用于衡量其达到最大可靠性的可能性，并与人工测试软件的可靠性进行比较。在这两种情况下，都假设系统中的风险率是随时间非线性递减的函数。

### *5.1.1.* 风险模型相关的界

**定义 2.** 对于任意人工测试的软件系统，以及任意 $K > 0$，$t > 0$，将随时间呈非线性递减的危害定义为：

$$z_{nld}(t) = \frac{K}{\sqrt{t}}, \text{ for some } K > 0, \text{and time } t > 0 \quad (29)$$

现在，以下推论定义了随机变量 $X$ 低于数值 $z_{nld}(t)$ 的偏差。

**推论 1.** 设 $X_1, X_2, \ldots, X_l$ 为独立伯努利试验，使得对于 $1 \le i \le l$, $Pr[X_i = 1] = p$，其中 $0 < p < 1$。那么，对于 $X = \sum_{i=1}^{l} X_i$, $\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} p = lp$，以及 $\exists$ 参数 $K > 0$、时间 $t > 0$ 和非线性递减风险率 $\frac{K}{\sqrt{t}}$：

$$Pr[X < \frac{K}{\sqrt{t}}] < e^{\frac{-(\sqrt{t}\mu_{\hat{z}} - K)^2}{2\mu_{\hat{z}} t}} \quad (30)$$

证明。 将 $z(t)$替换为 $z_{nld}(t)$（来自公式29）中的值，代入定理1，即可得到：

$$Pr[X < \frac{K}{\sqrt{t}}] < e^{\frac{-\mu_{\hat{z}}\left[1 - \frac{K}{\sqrt{t}\mu_{\hat{z}}}\right]^2}{2}} \quad (31)$$

化简上述公式即可证明该推论。

□

推论 1 表明，对于非线性递减风险模型，使用软件缺陷预测（SDP）的软件中发生的危害数量少于同一软件经人工测试后发生的危害总数的概率，其上界为最大值 of $e^{\frac{-(\sqrt{t}\mu_{\hat{z}} - K)^2}{2\mu_{\hat{z}} t}}$.

### *5.1.2.* 可靠性模型的界

本节首先定义遵循形式为 $\frac{K}{\sqrt{t}}$ 的非线性递减风险模型的软件的可靠性，然后确定其界。

**引理 3.** 对于风险模型 $z_{nld}(t) = \frac{K}{\sqrt{t}}$ 的非线性递减函数，在某个 $K > 0$ 和时间 $t > 0$ 下，其可靠性为：

$$R_{nld}(t) = e^{-2K\sqrt{t}} \quad (32)$$

证明。 我们假设软件中的危害是时间的非线性递减函数。其形式如下：

$$z_{nld}(t) = \frac{K}{\sqrt{t}}, \text{for some } K > 0, \text{and time } t > 0$$

现在，将 $z_{nld}(t)$ 代入 $R(t)$（在方程10中）将得到：

$$R_{nld}(t) = e^{-\int_0^t \frac{K}{\sqrt{x}} dx} \quad (33)$$

现在，化简方程 33 即可得到根据风险模型的非线性递减函数估算的软件可靠性。

□

**推论 2.** 设 $X_1, X_2, \ldots, X_l$ 为相互独立的伯努利 $i$ 试验，使得 $for,\ 1 \le i \le l,\ Pr[X_i = 1] = p, where$ $0 < p < 1$。那么对于 $X = \sum_{i=1}^{l} X_i, \mu_{\hat{R}} < e^{lp(e^{-t}-1)}$，参数 $K > 0$、时间 $t > 0$，以及具有非线性递减风险率的软件的可靠性，$R_{nld}(t) = e^{-2K\sqrt{t}}$:

$$Pr\left[e^{-Xt} > e^{-2K\sqrt{t}}\right] < e^{\frac{-e^{lp(1-e^{-t})}}{2}\left[e^{lp(e^{-t}-1)} - \frac{2K}{\sqrt{t}}\right]^2} \quad (34)$$

证明。 与定理 2类似，这个上尾的证明与下尾的证明非常相似，正如我们在推论1中所见。与之前一样，

$$\begin{aligned} Pr\left[e^{-Xt} > e^{-2K\sqrt{t}}\right] &= Pr\left[-Xt > -2K\sqrt{t}\right] \\ &= Pr\left[-X > -\frac{2K}{\sqrt{t}}\right] = Pr\left[X < \frac{2K}{\sqrt{t}}\right] \end{aligned} \quad (35)$$

在此，我们希望获得一个紧下界，表示随机变量 $X$ 与数值 $\frac{2K}{\sqrt{t}}$ 相差甚远的程度。这里，对于

某些 $K > 0$和 $t > 0$，数值 $\frac{2K}{\sqrt{t}}$ 被假设低于期望$\mu_{\hat{R}}$，在给定的时间段 $[0, t]$中。现在，将 $(1-\delta)\mu$ 在15 中的 $\frac{2K}{\sqrt{t}}$ 相等，则有：

$$(1-\delta)\mu = \frac{2K}{\sqrt{t}}$$

$$\Rightarrow \delta = 1 - \frac{2K}{\mu\sqrt{t}} \tag{36}$$

由于我们要寻找随机变量（$e^{-Xt}$）相对于人工测试软件可靠性的偏差的紧上界（最终将其转化为寻找另一种形式的紧下界），因此在上面的方程中，我们将用 $\mu$ 替换为使用 SDP 模型测试的软件的期望可靠性 $\mu_{\hat{R}}$。现在，将 $\delta$ （由方程 36得出）代入方程 15，以求得随机变量 $e^{-Xt}$ 偏离数值 $e^{-2K\sqrt{t}}$的紧上界。结果如下：

$$Pr\left[X < \frac{2K}{\sqrt{t}}\right] < e^{\frac{-\mu_{\hat{R}}\left[1-\frac{2K}{\sqrt{t}\mu_{\hat{R}}}\right]^2}{2}} \tag{37}$$

下方方程是对上方方程的简化。

$$Pr\left[X < \frac{2K}{\sqrt{t}}\right] = Pr\left[e^{-Xt} > e^{-2K\sqrt{t}}\right] < e^{-\frac{1}{2\mu_{\hat{R}}}\left[\mu_{\hat{R}}-\frac{2K}{\sqrt{t}}\right]^2} \tag{38}$$

将期望可靠性 $\mu_{\hat{R}} = e^{lp(e^{-t}-1)}$ 代入上面的方程，即可完成证明。 □

推论 2 表明，就可靠性（根据非线性递减风险模型计算）而言，使用 SDP 模型测试的软件比由人工测试人员测试的同一软件获得更高可靠性的概率受到最大值的限制

of $e^{\frac{-e^{lp(1-e^{-t})}}{2}\left[e^{lp(e^{-t}-1)}-\frac{2K}{\sqrt{t}}\right]^2}$.

## 5.2. 线性递减风险模型

假设软件中的危害以线性方式被检测并移除。虽然这种情况很少见，但我们仍给出线性递减风险模型的证明（分别从风险率和可靠性的角度）。最简单的危害线性递减函数假定为直线的斜截式方程；当直线的斜率为负时，该方程便成为时间的线性递减函数 $t$。

在第5.2.1节中，我们证明：与人工测试的软件相比，使用软件缺陷预测的软件有可能实现最小危害。类似地，在第5.2.2 节中，我们证明：与人工测试的软件的可靠性相比，使用软件缺陷预测的软件有可能实现最大可靠性。在这两种情况下，系统中的风险率均假设为随时间线性递减的函数。

### *5.2.1.* 风险模型界限

**定义3**。 对于任何人工测试的软件系统以及任意的 $K > 0$ 和 $m > 0$，遵循随时间线性递减函数的危害定义如下：

$$z_{ld}(t) = K - mt, \text{ for some } K > 0, m > 0 \text{ and time } t > 0 \tag{39}$$

现在，以下推论定义了随机变量 $X$ 低于数值 $z_{ld}(t)$ 的偏差。

**推论 3.** 设 $X_1$、$X_2$、…、$X_l$ 是相互独立的伯努利 $i$ 试验，使得对于 $1 \le i \le l$、$Pr[X_i = 1] = p$，其中 $0 < p < 1$。那么对于 $X = \sum_{i=1}^{l} X_i$、$\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} p = lp$，以及参数 $\exists$、$K > 0$、$m > 0$、时间 $t > 0$，对于线性递减风险率 $K - mt$：

$$Pr[X < K - mt] < e^{\frac{-(\mu_{\hat{z}}-K+mt)^2}{2\mu_{\hat{z}}}} \tag{40}$$

证明。 将 $z(t)$替换为 $z_{ld}(t)$（来自公式39）中的定理1，然后得到：

$$Pr[X < K - mt] < e^{\frac{-\mu_{\hat{z}}\left[1-\frac{K-mt}{\mu_{\hat{z}}}\right]^2}{2}} \tag{41}$$

经过几步简化，即可推导出随机变量 $X$ 低于数值 $z_{ld}(t)$ 的偏差紧界。 □

推论 3 表明，对于线性递减的风险模型，在使用软件缺陷预测的软件中，危害发生数量少于同一软件中危害总发生数量的概率（该软件由人工测试）

受的最大值所界定： $e^{\frac{-(\mu_{\hat{z}}-K+mt)^2}{2\mu_{\hat{z}}}}$.

### *5.2.2.* 可靠性模型方面的界

在以下引理中，我们定义软件的可靠性，该可靠性根据风险率线性递减函数计算得出。

**引理 4.** 对于风险模型 $z_{ld}(t) = K - mt$的线性递减函数，在某些 $K > 0$、$m > 0$和时间 $t$ >0的条件下，其可靠性为：

$$R_{ld}(t) = e^{\left[\frac{mt^2}{2} - Kt\right]} \tag{42}$$

证明。 我们假设手动测试的软件中的危害是时间的线性递减函数。其表达式为：

$$z_{ld}(t) = K - mt, \text{ for some } K > 0, m > 0 \text{ and time } t > 0$$

现在，将 $z_{ld}(t)$ 代入 $R(t)$ （由公式 10）可得：

$$R_{ld}(t) = e^{-\int_0^t K - mx\, dx} \tag{43}$$

现在，化简方程43 即可得证。 □

**推论 4.** 设 $X_1$、$X_2$、…、$X_l$ 为独立伯努利试验，使得对于 $1 \leq i \leq l$、$Pr[X_i = 1] = p$，其中$0 < p < 1$。那么，对于 $X = \sum_{i=1}^{l} X_i$、$\mu_{\hat{R}} < e^{lp(e^{-t}-1)}$，以及参数 $K > 0$、$m > 0$、时间 $t > 0$，对于具有线性递减风险率的软件的可靠性，

$R_{ld}(t) = e^{\left[\frac{mt^2}{2} - Kt\right]}$:

$$Pr\left[e^{-Xt} > e^{\left[\frac{mt^2}{2} - Kt\right]}\right] < e^{\frac{-e^{lp(1-e^{-t})}[2e^{lp(e^{-t}-1)} - 2K + mt]^2}{8}} \tag{44}$$

证明。这个上尾的证明与下尾的证明非常相似，正如推论3中所见。与之前一样，

$$Pr\left[e^{-Xt} > e^{\left[\frac{mt^2}{2} - Kt\right]}\right] = Pr\left[Xt < Kt - \frac{mt^2}{2}\right] = Pr\left[X < K - \frac{mt}{2}\right] \tag{45}$$

这里，我们希望得到一个紧下界，表示随机变量 $X$偏离数值 $K - \frac{mt}{2}$ 很远。这里，对于某个 $K > 0$，$m > 0$ 和 $t > 0$，假设数值 $K - \frac{mt}{2}$ 在给定时间段 $[0, t]$内低于期望$\mu_{\hat{R}}$。现在，将 $(1 - \delta)\mu$ 在15 中的值与 $K - \frac{mt}{2}$（在45中）相等，则有：

$$(1 - \delta)\mu = K - \frac{mt}{2}$$

$$\Rightarrow \delta = 1 - \frac{2K - mt}{2\mu} \tag{46}$$

由于我们要寻找随机变量（$e^{-Xt}$）相对于人工测试软件可靠性的偏差紧上界（最终将其转化为寻找另一种形式的紧下界），因此在上面的方程中，我们将用通过SDP模型测试的软件的期望可靠性$\mu_{\hat{R}}$ 替换 $\mu$ 。现在，将 $\delta$ （由方程46得出）代入方程15，以求得随机变量 $X$ 远离数值$K - \frac{mt}{2}$ 的偏差紧上界。如下所示：

$$Pr\left[X < K - \frac{mt}{2}\right] < e^{\frac{-\mu_{\hat{R}}\left[1 - \frac{2K-mt}{2\mu_{\hat{R}}}\right]^2}{2}} \tag{47}$$

经过几步简化即可得到以下证明：

$$Pr\left[e^{-Xt} > e^{\left[\frac{mt^2}{2} - Kt\right]}\right] < e^{-\frac{[2\mu_{\hat{R}} - 2K + mt]^2}{8\mu_{\hat{R}}}} \tag{48}$$

现在将期望可靠性 $\mu_{\hat{R}} = e^{lp(e^{-t}-1)}$ 代入上述方程，即可满足推论。

□

推论 4提供了证据表明，就可靠性而言（该可靠性根据非线性递减危害模型计算得出），使用SDP模型测试的软件比人工测试人员测试的相同软件获得更高可靠性的概率，其上限为 $e$的最大值 $\frac{-e^{lp(1-e^{-t})}[2e^{lp(e^{-t}-1)} - 2K + mt]^2}{8}$.

## 5.3. 非线性递增危害模型

由于级联效应，系统中的危害可能会增加 [20]。在这种情况下，一个模块的失效会导致后续模块失效。这会使软件系统中的危害以非线性（实际上甚至是指数级）的方式增加。能够提出的最简单的非线性递增风险模型，是一种危害随时间以二次方式增加的模型。以下小节给出了基于遵循非线性递增风险模型的风险模型所得的证明。

在第 5.3.1 节中，我们证明了使用SDP的软件有可能实现最小危害，相较之下，人工测试的软件所面临的危害更高。同样，在第 5.3.2 节中，我们证明了使用SDP的软件有可能实现最大可靠性，相较之下，人工测试的软件可靠性更低。

### *5.3.1.* 风险模型界限

**定义 4.** 对于任何经过人工测试的软件系统，以及任意 $K > 0$，将遵循时间非线性递增函数的危害定义为：

$$z_{nli}(t) = Kt^2, \text{ for some } K > 0 \text{ and time } t > 0 \tag{49}$$

与其他补充部分类似，公式 49 是威布尔风险率的一个特殊情形：当$m = 2$取该值时，风险率成为时间的二次函数。现在，以下推论定义了随机变量 $X$ 低于 $Kt^2$ 时的偏差。

**推论 5.** 设 $X_1$、$X_2$、…、$X_l$ 为独立伯努利试验，使得对于 $1 \leq i \leq l$，$Pr[X_i = 1] = p$，其中 $0 < p < 1$。那么，对于 $X = \sum_{i=1}^{l} X_i$、$\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} p = lp$、∃ 参数 $K > 0$、时间 $t > 0$，以及非线性递增的风险率 $z_{nli} = Kt^2$:

$$Pr[X < Kt^2] < e^{\frac{-(\mu_{\hat{z}} - Kt^2)^2}{2\mu_{\hat{z}}}} \tag{50}$$

证明。将 $z(t)$的值替换为 $z_{nli}(t)$（来自方程 49），代入定理 1，可得：

$$Pr[X < Kt^2] < e^{\frac{-\mu_{\hat{z}}\left[1 - \frac{Kt^2}{\mu_{\hat{z}}}\right]^2}{2}} \tag{51}$$

对上述方程进行化简即可完成证明。

□

推论 5 表明，就非线性递增风险模型而言，使用SDP 的软件中危害发生次数少于同一软件中危害总发生次数的概率

由人工测试人员测试时，其上限为 $e$ 的最大值 $\frac{-(\mu_{\hat{z}} - Kt^2)^2}{2\mu_{\hat{z}}}$.

#### *5.3.2.* 根据可靠性模型得出的界

在下述引理中，我们定义遵循风险率非线性递增模型的软件的可靠性。

**引理5**。 对于风险模型的非线性增长函数 $z_{nli}(t) = Kt^2$，在某个 $K > 0$和时间 $t$ >0下，其可靠性为：

$$R_{nli}(t) = e^{-\frac{Kt^3}{3}} \tag{52}$$

证明。 我们假设软件中的危害是时间的线性增长函数。其表达式为：

$$z_{nli}(t) = Kt^2, \text{ for some } K > 0 \text{ and time } t > 0$$

现在，将 $z_{nli}(t)$ 代入 $R(t)$ （由公式10）可得：

$$R_{nli}(t) = e^{-\int_0^t Kx^2 dx} \tag{53}$$

现在，化简公式53 可得具有非线性增长风险率函数的软件的可靠性。

□

**推论 6.** 设 $X_1$、 $X_2$、 …、 $X_l$ 为独立伯努利试验，使得对于， $1 \le i \le l$、 $Pr[X_i = 1] = p$，其中$0 < p < 1$。则对于 $X = \sum_{i=1}^{l} X_i$、 $\mu_{\hat{R}} < e^{lp(e^{-t}-1)}$ 和 ∃参数 $K > 0$、时间 $t > 0$，以及风险率非线性递增的软件的可靠性，$R_{nli}(t) = e^{\left[-\frac{Kt^3}{3}\right]}$：

$$Pr\left[e^{-Xt} > e^{\left[-\frac{Kt^3}{3}\right]}\right] < e^{\frac{-e^{lp(1-e^{-t})}\left[3e^{lp(e^{-t}-1)}-Kt^2\right]^2}{18}} \tag{54}$$

证明。 正如推论 5 中所见，上尾的证明与下尾的证明非常相似。和之前一样，

$$Pr\left[e^{-Xt} > e^{\left[-\frac{Kt^3}{3}\right]}\right] = Pr\left[Xt < \frac{Kt^3}{3}\right] = Pr\left[X < \frac{Kt^2}{3}\right] \tag{55}$$

现在，我们希望得到一个紧下界，以确定随机变量 $X$ 与数值 $\frac{Kt^2}{3}$ 相差很远。这里，对于某些$K > 0$和 $t > 0$，假定数值 $\frac{Kt^2}{3}$ 低于给定时间段 $\mu_{\hat{R}}$ 内的期望 $[0, t]$。现在，将 $(1-\delta)\mu$ 在 15 中的值与 $\frac{Kt^2}{3}$ 在 55中的值相等，则有：

$$(1-\delta)\mu = \frac{Kt^2}{3}$$

$$\Rightarrow \delta = 1 - \frac{Kt^2}{3\mu} \tag{56}$$

由于我们要找出随机变量（$e^{-Xt}$）偏离人工测试的软件的可靠性，因此在上述方程中，我们将 $\mu$ 替换为使用 SDP模型测试的软件的期望可靠性$\mu_{\hat{R}}$。现在，将 $\delta$ （来自方程56）中的值代入方程15，以求出随机变量偏离 $X$ 数值 $\frac{Kt^2}{3}$ 的紧上界。如下所示：

$$Pr\left[X < \frac{Kt^2}{3}\right] < e^{\frac{-\mu_{\hat{R}}\left[1-\frac{Kt^2}{3\mu_{\hat{R}}}\right]^2}{2}} \tag{57}$$

经过几步简化可得：

$$Pr\left[e^{-Xt} > e^{\left[-\frac{Kt^3}{3}\right]}\right] < e^{-\frac{[3\mu_{\hat{R}}-Kt^2]^2}{18\mu_{\hat{R}}}} \tag{58}$$

将期望可靠性 $\mu_{\hat{R}} = e^{lp(e^{-t}-1)}$ 代入上述方程即可完成证明。 □

推论6 表明，对于根据非线性递增风险模型计算的软件可靠性而言，使用SDP模型测试的软件获得更高可靠性的概率，其上限为使用人工测试的同一软件可靠性最大值的 mans 受 $e$ 的最大值限制 $\frac{-e^{lp(1-e^{-t})}\left[3e^{lp(e^{-t}-1)}-Kt^2\right]^2}{18}$.

### 5.4. 线性递增风险模型

通常，当系统发生磨损、劣化或功能升级时，危害会随着时间推移而增加 [21, 26]。例如，当系统发生功能升级时，由于软件功能得到增强，软件的复杂性可能会增加。因此，由于意外引入有缺陷的模块，系统很可能会出现危害 [20]。可以提出的最简单的递增风险模型，是假设系统中的危害随时间线性增加。以下各小节将证明基于线性递增风险模型的风险模型所得出的结论。

在第5.4.1节中，我们证明，与人工测试的软件中的危害相比，使用软件缺陷预测（SDP）的软件有可能实现最小危害。同样，在第5.4.2 节中，我们证明，与人工测试的软件的可靠性相比，使用 SDP 的软件有可能实现最大可靠性。

#### *5.4.1.* 风险模型界限

**定义5.** 对于任何人工测试的软件系统，以及任何 $K > 0$，随时间呈线性增长函数变化的危害定义如下：

$$z_{li}(t) = Kt, \text{ for some } K > 0 \text{ and time } t > 0 \tag{59}$$

现在，以下推论定义了随机变量 $X$ 低于值 $Kt$时的偏差。

**推论7**。 设 $X_1$、 $X_2$、 …、 $X_l$ 为独立伯努利试验，使得对于 $1 \leq i \leq l$、 $Pr[X_i = 1] = p$，其中 $0 < p < 1$。则对于 $X = \sum_{i=1}^{l} X_i$、 $\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} p = lp$，以及 ∃ 参数 $K > 0$、时间 $t > 0$和线性递增风险率 $z_{li} = Kt$:

$$Pr[X < Kt] < e^{\frac{-(\mu_{\hat{z}} - Kt)^2}{2\mu_{\hat{z}}}} \tag{60}$$

证明。 将 $z(t)$ 的值代入 $z_{li}(t)$（来自方程 59）和定理 1，可得:

$$Pr[X < Kt] < e^{\frac{-\mu_{\hat{z}}\left[1 - \frac{Kt}{\mu_{\hat{z}}}\right]^2}{2}} \tag{61}$$

化简上式即可得到界的最终形式。

□

推论 7 表明，对于线性递增的危害模型，使用软件缺陷预测的软件中危害发生次数少于由人工测试的同一软件中危害总发生次数的概率，其上界为 $e$ 的最大值 $\frac{-(\mu_{\hat{z}} - Kt)^2}{2\mu_{\hat{z}}}$。如果指数中的分子趋于无穷大，则可以得到更紧的界。

### *5.4.2.* 可靠性模型方面的界

**引理 6.** 对于风险模型 $z_{li}(t) = Kt$ 的线性增长函数，在某个 $K > 0$ 和时间 $t > 0$ 下，其可靠性为:

$$R_{li}(t) = e^{-\frac{Kt^2}{2}} \tag{62}$$

证明。 我们假设软件中的危害是时间的线性增长函数。其表达式为:

$$z_{li}(t) = Kt, \text{ for some } K > 0 \text{ and time } t > 0$$

现在，将 $z_{li}(t)$ 代入 $R(t)$ （来自公式 10）可得:

$$R_{li}(t) = e^{-\int_0^t K x dx} \tag{63}$$

现在，化简公式63即可得到软件的可靠性。

□

**推论 8.** 设 $X_1, X_2, \ldots, X_l$ 为独立伯努利试验，使得对于，$1 \leq i \leq l$, $Pr[X_i = 1] = p$，其中$0 < p < 1$。那么对于 $X = \sum_{i=1}^{l} X_i, \mu_{\hat{R}} < e^{lp(e^{-t}-1)}$，以及∃ 参数 $K > 0$、时间 $t > 0$，对于服从线性递增风险率的软件的可靠性，$R_{li}(t) = e^{-\frac{Kt^2}{2}}$:

$$Pr\left[e^{-Xt} > e^{-\frac{Kt^2}{2}}\right] = Pr\left[X < \frac{Kt}{2}\right] < e^{-\frac{[2e^{lp(e^{-t}-1)} - Kt]^2}{8e^{lp(e^{-t}-1)}}} \tag{64}$$

证明. 这个上尾的证明与下尾的证明非常相似，正如我们在推论7中看到的那样。

$$Pr\left[e^{-Xt} > e^{-\frac{Kt^2}{2}}\right] = Pr\left[Xt < \frac{Kt^2}{2}\right] = Pr\left[X < \frac{Kt}{2}\right] \tag{65}$$

现在，我们希望得到一个紧下界，以描述随机变量 $X$ 大幅偏离数值 $\frac{Kt}{2}$ 的情况。这里，对于某个$K > 0$ 和 $t > 0$, 假设数值 $\frac{Kt}{2}$ 低于期望 $\mu_{\hat{R}}$，且处于给定的时间段 $[0, t]$内。现在，将公式 $(1 - \delta)\mu$ 中的 15 与公式 $\frac{Kt}{2}$ 中的 65相等，则有:

$$(1 - \delta)\mu = \frac{Kt}{2}$$

$$\Rightarrow \delta = 1 - \frac{Kt}{2\mu} \tag{66}$$

由于我们要寻找随机变量（$e^{-Xt}$）相对于人工测试的软件的可靠性的偏差的紧上界（最终将其转化为寻找另一种形式的紧下界），因此在上述等式中，我们将用 $\mu$ 软件的期望可靠性 $\mu_{\hat{R}}$ 替换它，该软件是使用 SDP模型进行测试的。现在，将 $\delta$ （由方程 66得出）代入方程 15，以求得随机变量 $X$ 偏离数值 $\frac{Kt}{2}$ 的紧上界。结果如下:

$$Pr\left[X < \frac{Kt}{2}\right] < e^{\frac{-\mu_{\hat{R}}\left[1 - \frac{Kt}{2\mu_{\hat{R}}}\right]^2}{2}} \tag{67}$$

化简上述方程可得:

$$Pr\left[e^{-Xt} > e^{-\frac{Kt^2}{2}}\right] < e^{-\frac{[2\mu_{\hat{R}} - Kt]^2}{8\mu_{\hat{R}}}} \tag{68}$$

将期望可靠性 $\mu_{\hat{R}} = e^{lp(e^{-t}-1)}$ 代入上述方程，即可完成证明。 □

推论 8 提供了证据表明，对于根据线性递增风险模型计算的软件可靠性而言，使用 SDP模型测试的软件获得比同一软件经人工

mans 由 $e$ 的最大值限定 $\frac{[2e^{lp(e^{-t}-1)} - Kt]^2}{8e^{lp(e^{-t}-1)}}$.

## 5.5. 恒定风险模型

修订后的浴盆曲线的区域III（见图1）表示软件可靠性中的系统具有恒定风险率。也就是说，软件不像硬件那样进行功能升级。在这种情况下，系统可能会趋于过时，这表明升级系统的动力不足。因此，系统不会产生新的故障。于是，在无限时间段内，系统的风险率保持恒定 [26]。尽管软件不接受任何新的升级，但如果

软件正在使用中，就不会产生任何新的故障（假设软件始终包含固定数量的无害故障事件）。在这种情况下，系统的风险率将是某个任意常数，表示为 $\lambda$ [20] [26]。

在第5.5.1节中，我们证明：与人工测试的软件相比，使用软件缺陷预测的缺陷软件可以实现最小危害。同样，在第5.5.2节中，我们证明：与人工测试的软件的可靠性相比，使用软件缺陷预测的软件可以实现最大可靠性。在这两种情况下，我们都假设系统中的风险率为常数值，因此与时间无关。

### *5.5.1.* 危害模型的界

**定义6**。恒定危害模型对每个人工测试的软件系统定义如下：

$$z_c(t) = \lambda \tag{69}$$

现在，以下推论定义随机变量 $X$ 低于其期望 $\mu_{\hat{z}}$的偏差。

**推论9**。设 $X_1$、$X_2$、…、$X_l$ 为独立伯努利试验，使得对于 $1 \le i \le l$、$Pr[X_i = 1] = p$，其中 $0 < p < 1$。那么，对于 $X = \sum_{i=1}^{l} X_i$、$\mu_{\hat{z}} = \mathbb{E}[X] = \sum_{i=1}^{l} p = lp$，以及恒定风险率 $z_c = \lambda$：

$$Pr[X < \lambda] < e^{\frac{-(\mu_{\hat{z}}-\lambda)^2}{2\mu_{\hat{z}}}} \tag{70}$$

证明。 将 $z(t)$ 的值代入 $z_c(t)$中的定理 1，则可得：

$$Pr[X < \lambda] < e^{\frac{-\mu_{\hat{z}}\left[1-\frac{\lambda}{\mu_{\hat{z}}}\right]^2}{2}} \tag{71}$$

经过几步简化即可完成证明。

□

推论 9 表明，就恒定危害模型而言，使用软件缺陷预测的软件中发生的危害数量少于由人工测试人员测试的同一软件中的危害总数，其 $-(\mu_{\hat{z}}-\lambda)^2$受 $e$ $^{2\mu_{\hat{z}}}$ 的最大值界定。

### *5.5.2.* 可靠性模型的界

**引理7**。 对于恒定危害模型 $z_c(t) = \lambda$，其可靠性为：

$$R_c(t) = e^{-\lambda t} \tag{72}$$

证明。 我们假设在任意时刻， $t > 0$，软件中的危害数量为一个常数。其表示为：

$$z_c(t) = \lambda$$

现在，代入 $z_c(t)$ 到 $R(t)$ （来自公式10）可得：

$$R_c(t) = e^{-\int_0^t \lambda dx} \tag{73}$$

其中 $R_c(t)$ 是遵循恒定危害模型的软件的可靠性。现在，化简公式73即可得到采用恒定危害模型的软件的可靠性。

□

**推论10**。 设 $X_1, X_2, \ldots, X_l$ 为独立伯努利试验，使得 *for,* $1 \le i \le l$, $Pr[X_i = 1] = p$, 其中$0 < p < 1$。那么对于 $X = \sum_{i=1}^{l} X_i, \mu_{\hat{R}} < e^{lp(e^{-t}-1)}$，以及遵循恒定风险率的软件的可靠性， $R_c(t) = e^{-\lambda t}$：

$$Pr\left[e^{-Xt} > e^{-\lambda t}\right] < e^{\frac{-(e^{lp(e^{-t}-1)}-\lambda)^2}{2e^{lp(e^{-t}-1)}}} \tag{74}$$

证明。 这个上尾的证明与推论9中下尾的证明非常相似。根据引理 2 和 7，与之前一样，

$$Pr\left[e^{-Xt} > e^{-\lambda t}\right] = Pr\left[Xt < \lambda t\right] = Pr\left[X < \lambda\right] \tag{75}$$

证明可由推论9 得出，但此处使用的期望可靠性为 $\mu_{\hat{R}} = e^{lp(e^{-t}-1)}$。根据推论9的证明，我们有：

$$Pr[X < \lambda] < e^{\frac{-(\mu_{\hat{z}}-\lambda)^2}{2\mu_{\hat{z}}}}$$

在上述公式中用期望可靠性 $\mu_{\hat{R}}$ 替换 $\mu_{\hat{z}}$，即可得到紧下界。

□

推论 10 表明，就根据恒定危害模型计算的软件可靠性而言，与由人工测试人员测试的软件相比，使用软件缺陷预测模型测试的软件获得更高可靠性的概率受 $e$ 的最大值约束 $\frac{-(e^{lp(e^{-t}-1)}-\lambda)^2}{2e^{lp(e^{-t}-1)}}$。

## 6. 讨论

第 4 节和第 5 节中的证明基于这样的假设：随机变量的取值属于集合 {0,1}，这表示任何有缺陷的组件（一个模块）若被预测为无缺陷，都可能导致系统发生一次失效。基于这一假设，在第 6.1 节中，我们讨论对随机变量定义进行合理修订的可能性。具体而言，如果每个休眠的有缺陷模块在软件系统中产生的不是一次失效，而是产生某种特定形式的威布尔分布危害，我们就会给出一些额外的界。

此外，在第 4 节和第 5节中，我们针对随机变量的偏差提供了各种证明。该随机变量（以使用 SDP模型测试的软件的风险率和可靠性表示）偏离了实际值（以人工测试的软件的风险率和可靠性表示）。现在，在第6.2节中，我们结合构造证明时使用的参数来分析这些定理。

### 6.1. 关于随机变量的定义

第 4 节和第 5 节中给出的所有证明都基于这样一个假设：随机变量的取值为 1 或 0。换言之，如果随机变量取值为 1，则表示一个假阴性模块（其中包含一个休眠缺陷）可能导致系统发生一次失效。在此基础上，本节将提供推导多种证明的依据，这些证明基于这样的假设：随机变量的取值由参数的函数决定，例如 $\hat{K}$、$\hat{m}$和 $t$，而非仅取值为 1 或 0。以下假设可确保含义不变：

**假设10**。 在任何软件中，每个误分类缺陷模块都服从风险形式为 $\hat{K}t^{\hat{m}}$的威布尔分布。

综合假设 10，我们计算软件系统的风险率，该风险率由各个组件（软件模块）的风险率求和得到。形式上，对于同一个新开发的软件 $S$，它包含 $n$ 个模块，定义随机变量 $Y$，当模块被错误地分类为干净类时，取值为 $Kt^m$；当模块 $M_i$ 被正确地分类为干净类时，取值为0。定义如下：

$$Y_i = \begin{cases} \hat{K}t^{\hat{m}}, & \text{if } M_i \text{ is classified wrongly into clean class.} \\ 0, & \text{if } M_i \text{ is classified correctly into clean class.} \end{cases} \tag{76}$$

这里，我们假设每个误分类的有缺陷软件模块都服从类似类型的分布。在方程 76中，对于这些参数 $K > 0$、$\hat{m} > -1$和 $t > 0$，值 $Kt^m$表示随机变量 $Y_i$所服从的威布尔分布。

根据假设 2，由于我们已设定某个组织会使用传统软件缺陷预测模型（该模型通过批量学习开发）提供的服务；并且根据假设 3，我们确保每个新开发模块的预测值具有相同的概率，因此，与定义随机变量 $X_i$的概率值时类似，在此情况下，每个被预测为干净但被错误归类的模块都被赋予概率 $p$。表示如下：

$$\mathbf{Pr}[Y_i = \hat{K}t^{\hat{m}}] = p, \text{ and, } \mathbf{Pr}[Y_i = 0] = 1-p, \text{ for } 1 \le i \le l. \tag{77}$$

现在，我们估算软件中可能存在的危害实例总数，$P$。与随机变量 $X$的情况类似，由于所有模块都是随机且独立地进行测试的，因此我们将每一对预测视为相互独立。此外，我们假设假设1-3和5-9必须成立，才能定义以下随机变量 $Y$。现在，软件中可能存在的危害实例总数（$Y$）估算如下：

$$Y = \sum_{i=1}^{l} Y_i \tag{78}$$

这里，随机变量 $Y$ 表示软件中可能存在的危害实例总数，$P$。公式78中推导出的随机变量实际上就是基于 SDP 的软件中的风险率，该软件满足假设10。现在，设$\hat{z}_Y(t)$ 为基于 SDP 的软件的风险率，则有：

$$\hat{z}_Y(t) = Y \tag{79}$$

与随机变量 $X_i$ 的定义类似，由于 SDP模型以相同的概率 $p$ 独立地预测每个新开发软件模块的类别标签，$Y_i$ 成为一次伯努利试验。现在，对于相互独立的伯努利随机变量 $Y_1$、$Y_2$、⋯、$Y_l$，它们的和（$Y$）服从二项分布。下面，我们推导整个软件中可能发生的危害数量的期望值。二项分布的均值推导如下（根据期望的线性性）：

$$\begin{aligned} \mu_{\hat{z}_Y} &= \mathbb{E}[Y] = E\Big[\sum_{i=1}^{l} Y_i\Big] = \sum_{i=1}^{l} \mathbb{E}[Y_i] \\ &= \sum_{i=1}^{l} \left[[\hat{K}t^{\hat{m}} * \mathbf{Pr}[Y_i = \hat{K}t^{\hat{m}}] + 0 * \mathbf{Pr}[Y_i = 0]\right] \\ &= \sum_{i=1}^{l} p\hat{K}t^{\hat{m}} = lp\hat{K}t^{\hat{m}} \end{aligned} \tag{80}$$

危害实例的预期数量（即 $\mu_{\hat{z}_Y}$）在方程 80 中，不过是软件中危害实例的威布尔分布的缩放形式 $P$。在这种情况下，每个随机变量 $Y_i$ 以概率 $Kt^m$ 取值 $p$，并以概率 $1-p$取值为 0。在方程 80中，我们依据期望的线性性进行推导，因为随机变量 $Y_i$和 $i \in \{1, 2, \cdots, l\}$ 被假定彼此独立。现在，在以下小节中，我们将基于软件的风险率（和可靠性）推导紧界。

#### *6.1.1.* 根据软件风险率确定的紧界

现在，利用方程 78和 80中的事实，以及定义 1，我们将给出以下结论：与人工测试的软件相比，使用SDP模型实现的软件更有可能达到更少的危害。

**定理3.** 设 $Y_1, Y_2, \ldots, Y_l$ 为相互独立的伯努利试验，使得对于，$1 \le i \le l$, $Pr[Y_i = Kt^m] = p$，其中$0 < p < 1$。则 $\exists$ 参数 $K > 0, m > -1, \hat{K}, \hat{m}$，时间 $t > 0$。则对于 $Y = \sum_{i=1}^{l} Y_i, \mu_{\hat{z}_Y} = \mathbb{E}[Y] = lp\hat{K}t^{\hat{m}}$，以及威布尔风险率，$Kt^m$：

$$Pr[Y < Kt^m] < e^{\frac{-[lp\hat{K}t^{\hat{m}} - Kt^m]^2}{2lp\hat{K}t^{\hat{m}}}} \tag{81}$$

证明。 由公式80可知，危害的期望数量（假设这些危害是从休眠的有缺陷模块中观测到的，且每个休眠缺陷都可能导致风险率呈某种特定形式的威布尔分布）

在给定时间段 $[0,t]$ 内，使用 SDP 模型的软件系统中的危害数量：

$$\mu_{\hat{z}_Y} = lp\hat{K}t^{\hat{m}}$$

我们知道，对于某些 $0 < \delta \leq 1$，以及 $\mu$，根据公式15，使用切尔诺夫界（独立伯努利试验之和的下尾界），$Y$，可得：

$$Pr[Y < (1-\delta)\mu] < e^{\frac{-\mu\delta^2}{2}}$$

现在，我们希望得到一个紧下界，以衡量随机变量 $Y$ 在给定时间段 $[0,t]$内偏离人工测试软件风险率 $Kt^m$的程度。令公式15 中的 $(1-\delta)\mu$等于 $z(t)$在11中的值，则有：

$$(1-\delta)\mu = Kt^m$$

由于我们正在计算随机变量 $Y$偏差的可能紧界，因此用$\mu_{\hat{z}_Y}$表示危害的期望数量，而不使用 $\mu$。现在，将$\mu_{\hat{z}_Y}$（根据公式80）代入上式，可得：

$$\Rightarrow \delta = 1 - \frac{Kt^m}{lp\hat{K}t^{\hat{m}}} \tag{82}$$

此处，$\delta$ 定义了相对于期望值$\mu_{\hat{z}_Y}$的左侧边界值。

由于假设（经人工测试的）软件系统的风险率服从 Weibull 分布，因此可利用公式 $Kt^m$ （其中 $Kt^m$ 是期望值的缩放值，$\mu_{\hat{z}_Y}$）推导出（使用 SDP 模型测试的）软件系统的风险率低于期望值的概率的紧下界，公式编号为 15，如下：

$$Pr[Y < Kt^m] < e^{\frac{-lp\hat{K}t^{\hat{m}}\left[1-\frac{Kt^m}{lp\hat{K}t^{\hat{m}}}\right]^2}{2}} \tag{83}$$

在上述公式中，对于某些 $K > 0$，$m > -1$，$K > 0$，$\hat{m} > -1$，以及 $t > 0$，假设值 $Kt^m$ 低于期望值 $lp\hat{K}t^{\hat{m}}$。接下来，经过几步化简即可证明该定理。□

定理 3 表明，使用 SDP 的软件中危害发生次数较少的概率，低于同一软件经人工测试后出现的危害总数，且其上界为最大值 $e^{\frac{-[lp\hat{K}t^{\hat{m}}-Kt^m]^2}{2lp\hat{K}t^{\hat{m}}}}$。如前所述，此处的证明基于每个随机变量 $Y_i$ 取自特定形式的威布尔分布。因此，定理1 与3之间的差异基于随机变量期望的定义。

### *6.1.2.* 软件可靠性方面的紧界

本节首先给出一个引理，用于根据软件中的总危害计算软件的可靠性（根据公式78，可表示为 $Y$，在任意时刻，$t > 0$），其中该软件采用SDP模型进行测试。随后，我们给出一个紧上界，用于界定随机变量（表示采用SDP模型测试的软件的可靠性）与人工测试的软件的可靠性之间的偏差。

**引理 8.** 设 $Y_1, Y_2, \ldots, Y_l$ 是独立伯努利试验，则对于软件系统中的危害 $Y = \sum_{i=1}^{l} Y_i$，其可靠性为：

$$\hat{R}_Y(t) = e^{-Yt} \tag{84}$$

证明。 我们知道，使用 SDP 进行测试的软件中的危害是随机变量 $Y$。然后，利用公式 10：

$$\hat{R}_Y(t) = e^{-\int_0^t Y dx} \tag{85}$$

这里，我们用 $\hat{R}_Y(t)$ 表示使用 SDP模型测试的软件的可靠性，其中每个误分类的缺陷模块都会使软件中的风险率呈现特定形式的威布尔分布。现在，简化公式 85即可得出基于 SDP 的软件的可靠性。

□

与危害的情况类似，为了求出随机变量（以使用SDP模型测试的软件的可靠性表示）偏离人工测试软件可靠性的紧界（也就是说，$e^{-Kt^{m+1}}$即 $e^{-Kt^{m+1}}$），我们需要计算使用 SDP模型测试的软件的期望可靠性。根据引理8，我们知道使用 SDP模型测试的软件的可靠性。现在，期望可靠性（$\mu_{\hat{R}_Y}$或 $\mathbb{E}[R_Y(t)]$）推导如下：

$$\mathbb{E}[\hat{R}_Y(t)] = \mu_{\hat{R}_Y} = \mathbb{E}[e^{-Yt}] = \mathbb{E}\left[e^{-t\sum_{i=1}^{l} Y_i}\right] \tag{86}$$

由于假设随机变量 $(Y_i)$ 相互独立，因此指数中的各项之和将转化为指数项的乘积。表示如下：

$$\mathbb{E}\left[e^{-t\sum_{i=1}^{l} Y_i}\right] = \prod_{i=1}^{l} \mathbb{E}\left[e^{-tY_i}\right] \tag{87}$$

这里，随机变量 $e^{-tY_i}$ 取值 $e^{\hat{K}t^{\hat{m}+1}}$ 的概率为 $p$，取值 1 的概率为 $1-p$。现在，利用这些事实，由方程 87可得：

$$\prod_{i=1}^{l} \mathbb{E}\left[e^{-tY_i}\right] = \prod_{i=1}^{l} \left[pe^{\hat{K}t^{\hat{m}+1}} + 1 - p\right] = \prod_{i=1}^{l} \left[p(e^{\hat{K}t^{\hat{m}+1}} - 1) + 1\right] \tag{88}$$

我们知道，$1+x < e^x$。现在，将此不等式与 $x = p(e^{\hat{K}t^{\hat{m}+1}} - 1)$结合，改写方程 88，得到期望可靠性：

$$\mathbb{E}[\hat{R}_Y(t)] = \mu_{\hat{R}_Y} = \prod_{i=1}^{l}\left[p(e^{\hat{K}t^{\hat{m}+1}} - 1) + 1\right] < \prod_{i=1}^{l} e^{p(e^{\hat{K}t^{\hat{m}+1}}-1)} = e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)} \quad (89)$$

上述不等式不会影响定理4中的最终界。现在利用以下引理1 和8，我们给出软件（使用 SDP 模型实现）相较于人工测试的软件达到最大可靠性的紧界。

**定理 4**。 设 $Y_1, Y_2, \ldots, Y_l$ 为独立伯努利试验，使得对于 $1 \le i \le l$, $Pr[Y_i = Kt^m] = p$，其中 $0 < p < 1$ $\exists$ $K > 0$ $m > -1$ $\hat{m} >$。那么参 $-1$ 数，，$\hat{K} > 0$，，$t > 0$ 以及时间 。那么对于 $Y = \sum_{i=1}^{l} Y_i$, $\mu_{\hat{z}_Y} = \mathbb{E}[\hat{R}_Y(t)] < e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}$，对于威布尔可靠性函数，$e^{\frac{-Kt^{m+1}}{m+1}}$：

$$Pr\left[e^{-Yt} > e^{\frac{-Kt^{m+1}}{m+1}}\right] < e^{\frac{-\left[e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)} - \frac{Kt^m}{m+1}\right]^2}{2e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}}} \quad (90)$$

证明。 正如我们在定理 3中所见，证明这个上尾与证明下尾非常相似。根据引理8和2，与之前一样，

$$Pr\left[e^{-Yt} > e^{\frac{-Kt^{m+1}}{m+1}}\right] = Pr\left[e^{Yt} < e^{\frac{Kt^{m+1}}{m+1}}\right] = Pr\left[Yt < \frac{Kt^{m+1}}{m+1}\right] = Pr\left[Y < \frac{Kt^m}{m+1}\right] \quad (91)$$

首先，我们希望为随机变量 $e^{-Yt}$偏离数值 $e^{\frac{-Kt^{m+1}}{m+1}}$ 很远的情况求得一个紧上界。现在，根据方程91，我们希望为随机变量 $Y$远低于数值 $\frac{Kt^m}{m+1}$的情况求得紧下界。根据方程 15和 91，对于某个 $K > 0$，$m > -1$，$\hat{K} > 0$，$\hat{m} > -1$，且 $t > 0$，假设数值 $\frac{Kt^m}{m+1}$ 低于期望 $e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}$。现在，将方程 $(1-\delta)\mu$ 15 中的 $\frac{Kt^m}{m+1}$ 与方程 91中的对应项相等，则有：

$$(1-\delta)\mu = \frac{Kt^m}{m+1} \quad (92)$$

由于我们正在计算随机变量 $Y$偏差的可能紧界，因此用 $\mu_{\hat{z}_Y}$表示危害的期望数量，以替代 $\mu$。现在，将 $\mu_{\hat{z}_Y}$（来自公式 80）代入公式 92，则有：

$$\Rightarrow \delta = 1 - \frac{Kt^m}{(m+1)e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}} \quad (93)$$

将 $\delta$（来自公式93）代入公式 15，则：

$$Pr\left[Y < \frac{Kt^m}{m+1}\right] < e^{\frac{e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}\left[1 - \frac{Kt^m}{(m+1)e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}}\right]^2}{2}} \quad (94)$$

经过几步简化即可完成该定理的证明。

□

定理4 表明，在使用SDP模型的软件中，获得更高可靠性的概率高于由人工测试人员测试的相同软件的可靠性，并且该概率受以下最大值的限制 $e^{\frac{-\left[e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)} - \frac{Kt^m}{m+1}\right]^2}{2e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}}}$。与定理3的结果类似，定理4 的证明基于每个随 $Y_i$ 机变量都取自威布尔分布的特定形式（随后，我们利用该风险模型计算可靠性）。因此，定理2 和4 之间的差异取决于随机变量期望的定义。

定理3 和4 是通过改变随机变量的定义得到的。根据定理3和4，类似于第5节提供的补充证明，我们可以推导出关于风险率和可靠性的众多证明。

### 6.2. 定理1、2、3和4

定理1和定理2中的界随以下因素变化：(1) 预测为干净模块的数量；(2) 观察风险值的时间。首先，由于概率（$p$）对于目标应用而言是固定值，因此假设参数取任意固定值（例如 $K, m$）（假设 $K$ 和$m$ 是根据某些准则得出的）[5]，，且时间为 $t$。与人工测试软件的风险率相比，使用 SDP模型测试的软件中观测到最小危害的概率在 $l$ 上呈指数级地小，这意味着预测为干净模块的数量越多，就越有可能显现真实情况。其次，假设参数 $K$ 和 $m$ 是根据人工测试软件中的某些准则得出的。随着时间推移（也就是说，对于较大的时间值 $t$），对于某些较大的参数值 $K$ 和 $m$（假设这些参数必须遵循由 $\delta$ 施加的条件 $\frac{-(lp-Kt^m)^2}{-e^{lp(1-e^{-t})}}$（即 $0 < \delta \le 1$）），项 $e^{\frac{}{2lp}}$（来自定理 1）和 $e^{\left[e^{lp(e^{-t}-1)} - \frac{Kt^m}{m+1}\right]^2}$（来自定理 2）将变得极小，甚至接近 0。因此，随着时间推移（假设我们已经观测到 $l$个干净模块），我们可以推导出无效性

[5]估算参数（例如 $K$，$m$）在任意时刻 $t$ 的值，对于构建证明至关重要。然而，在这项初步工作中，我们假设这些参数的值必须遵循 $\delta$ 所施加的条件（即 $0 < \delta \le 1$）。

在真实世界测试环境中，与人工测试的软件系统相比，SDP模型（以风险率和可靠性表示）的无效性。

由于第5 节中给出的推论是从定理1 和2推导出来的，因此，上述相同的分析也适用于描述这些界的行为。

与对定理1和定理2所作的分析类似，定理3和定理4中界的行为变化也可以根据以下因素来描述：（1）预测为无缺陷的模块数量；（2）观察风险值的时间。首先，与上述分析类似，由于概率（$p$）对于目标应用是一个固定值，因此假设在参数取任意固定值—— $K$、$m$、 $\hat{K}$、 $\hat{m}$[6]——且时间为 $t$时，与人工测试软件的风险率相比，使用SDP模型测试的软件中观察到最小危害的概率在 $l$上呈指数级小，甚至接近于零。这意味着，预测为无缺陷的模块数量较多时，我们更有可能看到所开发预测模型的真实情况。其次，假设参数 $K$ 和 $m$ 是根据人工测试软件所采用的某些标准得出的。随着时间推移（即时间 $t$取较大值时），对于参数 $K$、 $m$、 $\hat{K}$和 $\hat{m}$ 的一些较大取值（假设这些参数必须遵循 $\delta$ 所施加的条件，即$0 < \delta \leq 1$），项 $e^{-\left[e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}-\frac{Kt^m}{m+1}\right]^2}$ （来自定理3）和 $e^{2e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}}$（来自定理4）将变得极小，甚至接近于0。因此，随着时间流逝（假设我们已经观察了 $l$ 个干净模块），我们可以得出结论：SDP模型在真实世界测试环境中无效。

在上述分析中，我们使用一种特定形式的风险率来描述每个有缺陷模块如何导致系统故障。我们选用威布尔分布的特殊形式，只是为了描述随机变量可以取失效函数的任意形式，而不只是取 1 或 0。值得注意的是，在给出证明（定理 $\hat{K}$、 $\hat{m}$ 和3、4）之前，可以估计诸如$\hat{K}$，以及 $\hat{m}$ 之类的参数。

## 7. 结论

本研究旨在填补文献中的一项空白，探讨 SDP 模型在真实世界测试环境中的可行性。为了为二元缺陷预测模型的可行性提供理论证明，我们作出了一些假设。这些假设简化了证明过程中的计算。放宽上述假设可能会使我们观察到更精确的界。

为评估软件缺陷预测模型的可行性，我们首先根据测试集，将假阴性数占观测到的无缺陷样本总数的百分比作为概率值。假设任何最优分类器都会在目标项目中产生缺陷注入（也称为假阴性）和原始无缺陷样本（也称为真阴性），且每次缺陷注入都会导致系统发生一次失效，我们据此定义了目标软件项目中的故障分布。随后，我们计算了软件中因缺陷注入而产生的预期故障事件数。

针对在人工测试条件下测试的同一软件，我们利用其风险率及可靠性，采用著名的尾部不等式技术——切尔诺夫界，给出了紧界。我们的方法主要通过比较使用软件缺陷预测模型测试的软件与人工测试软件的风险率（及其可靠性），来给出这些界。我们假设人工测试的软件的风险率服从威布尔分布，由此得到紧界。这些界是动态的，可替换为任何现有的危险函数，以衡量软件缺陷预测模型的可行性。为说明使用各种危害模型的可能性，我们还基于威布尔分布危害模型的补充性质给出了一些证明。

就威布尔危害率而言，我们推导出了一个最大界 $e^{-(\mu_{\hat{z}}-Kt^m)^2}$，表示在使用软件缺陷预测的软件中观测到的危害少于由人工测试人员测试的同一软件中的危害总数的概率。类似地，就根据风险率的威布尔分布推导出的软件可靠性而言，我们推导出了一个最大界 $e^{-\left[e^{lp(e^{-t}-1)}-\frac{Kt^m}{m+1}\right]^2}$，表示使用软件缺陷预测的软件获得的可靠性高于由人工测试人员测试的同一软件可靠性的概率。对这些界的总结表明，当随机变量（建模为风险率或可靠性）显著偏离期望时，我们可以得出结论：缺陷预测模型在真实世界测试环境中效率低下。

在上述界中，我们假设每个误分类缺陷实例都会发生一次失效。除上述界外，我们还假设每个误分类缺陷实例会产生威布尔分布的风险率，而不是在系统中引发一次失效，并据此推导出了另外两个关于风险率和可靠性的界。就威布尔风险率而言，我们推导出了一个最大界 $e^{\frac{-[lp\hat{K}t^{\hat{m}}-Kt^m]^2}{2lp\hat{K}t^{\hat{m}}}}$ 表示观察到使用软件缺陷预测的软件中的危害数量少于由人工测试人员测试的同一软件中的危害总数的概率。同样，对于根据危害的威布尔分布推导出的软件可靠性，

[6]在任意时刻 $t$估计诸如 $K$、 $m$、 $\hat{K}$ 和 $\hat{m}$ 之类的参数，对于构建证明至关重要。不过，在这项初步研究中，我们假设这些参数的取值必须满足 $\delta$ 所施加的条件（即 $0 < \delta \leq 1$）。在所有这些参数中，$\hat{K}$ 和 $\hat{m}$ 用于描述误分类缺陷模块所引发的风险率。我们假设使用与威布尔分布相似的形式（即 $\hat{K}t^{\hat{m}}$），来描述采用 SDP模型的软件中的危害事件。由于威布尔分布能够理想地表示软件或软件组件（模块）的风险率，因此我们采用相同的表示形式来描述每个误分类缺陷模块的风险率。

$$\frac{-\left[e^{lp(e^{\hat{K}t^{\hat{m}+1}}-1)}-\frac{Kt^m}{m+1}\right]^2}{2\pi p(e^{\hat{K}t^{\hat{m}+1}}-1)\left[\frac{Kt^m}{m+1}\right]^2}$$

风险率，我们推导出了软件缺陷预测后，软件获得更高可靠性的概率，相较于由人工测试人员测试同一软件时的可靠性。基于这两个界，只需将威布尔风险函数替换为风险率的任何其他函数形式，即可推导出众多证明。

我们认为，在真实世界测试环境中对所开发的二分类模型（在我们的案例中，即软件缺陷预测模型）进行批判性评估，在机器学习研究中具有新颖性，并有望为其他应用的可行性提供见解。基于这一思路的一些潜在未来研究方向列于下一节。

### 7.1. 未来计划

定理1、2、3和4 为所开发的二分类模型（SDP模型）在真实世界测试环境中的可行性提供了初步界。在本研究的范围内，定理的扩展1、2、3和4 不胜枚举。以下是几个例子：

1. 在本研究中，为了给出证明，我们假设人工测试的软件具有威布尔模型的危害率。若在构造证明时采用最先进的危害（及可靠性）模型，而不是威布尔分布，定理1、2、3和4 给出的界将更具应用针对性。

2. 本研究假设随机变量之间相互独立（包括 $X_i$ 之间以及 $Y_i$ 之间）。相反，我们将基于随机变量之间存在依赖关系这一假设推导新的界，从而确保在软件系统发生级联故障时也能推导这些界。由于任何软件中模块之间都存在耦合这一事实，某个模块发生故障可能会导致与之耦合的模块发生故障 [20]。因此，软件可能因级联效应而经历更多故障 [20]。在这种情况下，随机变量之间的依赖关系体现了软件模块之间的耦合。因此，在假设随机变量之间存在依赖关系的情况下，研究缺陷预测模型在真实世界测试环境中的实用性，是一项亟需开展的研究。

我们希望这项工作能为一些潜在的未来研究方向奠定基础，也就是探讨这项工作的普适性。我们希望这项工作能开辟一条新途径，利用概率界分析预测模型（尤其是二分类模型）对目标应用（任何类型）的影响。应用示例包括：光学字符识别（OCR）系统、机器翻译系统、欺诈检测系统、肿瘤检测系统等。

## 参考文献

[1] B. T. Compton、C. Withrow，Ada 软件缺陷的预测与控制，《系统与软件杂志》12 (3) (1990) 199–207。 [2] T. M. Khoshgoftaar、J. C. Munson，利用软件复杂度度量指标预测软件开发错误，《IEEE通信领域精选期刊》8 (2) (1990) 253–261。[3] U. S. Bhutamapuram、R. Sadam，使用基于自助聚合的多样化集成学习技术进行项目内缺陷预测，《沙特国王大学学报》-计算机与信息科学 34 (10, Part A) (2022) 8675–8691。[4] V. R. Basili、L. C. Briand、W. L. Melo，对面向对象设计度量作为质量指标的验证，《IEEE软件工程汇刊》22 (10) (1996) 751–761。 [5] N. E. Fenton、M. Neil，对软件缺陷预测模型的批判，《IEEE软件工程汇刊》25 (5) (1999) 675–689。[6] V. U. B. Challagulla、F. B. Bastani、I.-L. Yen、R. A. Paul，基于机器学习的软件缺陷预测技术的实证评估，《国际人工智能工具期刊》17 (02) (2008) 389–400。[7] K. Gao、T. M. Khoshgoftaar，软件故障预测计数模型的综合实证研究，《IEEE可靠性汇刊》56 (2) (2007) 223–236。 [8] K. O. Elish、M. O. Elish，使用支持向量机预测易产生缺陷的软件模块，《系统与软件杂志》81 (5) (2008) 649–660。[9] C. Catal、B. Diri，软件故障预测研究的系统综述，《专家系统与应用》36 (4) (2009) 7346–7354。[10] E. Erturk、E. A. Sezer，采用混合方法进行迭代式软件故障预测，《应用软计算》49 (2016) 1020–1033。[11] C. Jin，基于领域自适应学习和优化的跨项目软件缺陷预测，《专家系统与应用》171 (2021) 114637。[12] U. S. B、R. Sadam，预测决策对软件项目的影响有多大？来自跨项目缺陷预测模型的成本、服务时间和失效分析，《系统与软件杂志》195 (1) (2023) 111522。doi:https://doi.org/10.1016/j.jss.2022.111522。URL https://www.sciencedirect.com/science/article/pii/S0164121222001984 [13] T. Zimmermann、N. Nagappan、H. Gall、E. Giger、B. Murphy，跨项目缺陷预测：关于数据、领域与过程的大规模实验，载于：第七届欧洲软件工程会议与美国计算机协会 SIGSOFT 软件工程基础研讨会联合会议论文集，美国计算机协会，荷兰阿姆斯特丹，2009，第 91–100 页。 [14] T. J. McCabe，一种复杂度度量，《IEEE软件工程汇刊》4 (1976) 308–320。[15] M. H. Halstead，《软件科学要素》（操作系统与编程系统系列），爱思唯尔科学出版社，纽约美洲大道 655 号，纽约州纽约市，美国，1977。[16] J. Nam、W. Fu、S. Kim、T. Menzies、L. Tan，异构缺陷预测，《IEEE软件工程汇刊》44 (9) (2017) 874–896。[17] S. Herbold、A. Trautsch、J. Grabowski，跨项目缺陷预测方法基准测试的比较研究，《IEEE软件工程汇刊》44 (9) (2017) 811–833。 [18] G. James、D. Witten、T. Hastie、R. Tibshirani，《统计学习导论》，第 112 卷，施普林格，施普林格纽约，纽约州，2013。 [19] S. Shalev-Shwartz、S. Ben-David，《理解机器学习：从理论到算法》，剑桥大学出版社，纽约州纽约市西 20 街 40 号，美国，2014。[20] M. R. Lyu 等，《软件可靠性工程手册》，第 222 卷，IEEE计算机学会出版社，加利福尼亚州洛斯阿拉米托斯，1996。 [21] R. S. Pressman，《软件工程：实践者的方法》，帕尔格雷夫·麦克米伦，纽约美洲大道 1221 号，2005。 [22] M. Hellman、J. Raviv，错误概率、疑义度与 Chernoff 界，《IEEE信息论汇刊》16 (4) (1970) 368–372。

[23] R. Motwani、P. Raghavan，《随机算法》，剑桥大学出版社，纽约州纽约市西 20 街 40 号，1995。[24] S. M. Ross，《概率模型导论》，学术出版社，荷兰阿姆斯特丹，2014。[25] C. M. Bishop、N. M. Nasrabadi，《模式识别与机器学习》，第 4 卷第 4 号，施普林格，施普林格纽约，纽约州，2006。[26] M. A. Hartz、E. L. Walker、D. Mahar，软件可靠性导论：现状综述，可靠性分析中心（RAC），纽约州罗马市米尔街 201 号，邮编 13440，美国，1997。URL https://books.google.co.in/books?id=DqecmgEACAAJ[27] S. Keene，硬件与软件可靠性的比较，《可靠性评论》14 (4) (1994) 5–7。[28] Y.-S. Huang、K.-C. Chiu、W.-M. Chen，一种针对不完美调试的软件可靠性增长模型，《系统与软件杂志》188 (2022) 111267。[29] S. S. Gokhale、T. Philip、P. N. Marinos、K. S. Trivedi，通过测试覆盖率统一有限故障非齐次泊松过程模型，载于：ISSRE'96 第七届国际软件可靠性工程研讨会论文集，IEEE，纽约州怀特普莱恩斯，美国，1996，第 299–307 页。[30] H. Chernoff，一种基于观测值之和的假设检验渐近效率度量，《数学统计年刊》23 (4) (1952) 493–507。