基于历史信息的自适应测试用例优先级技术

常龙辉[1,2] 缪淮扣[1,2] 肖 蕾[1]
(上海大学计算机工程与科学学院 上海 200444)[1] (上海市计算机软件评测重点实验室 上海 201112)[2]

摘 要 在软件迭代开发的过程中,测试用例优先级技术因能有效地提高回归测试的效率,降低时间开销和人力成本,受到研究者的广泛关注,许多优化方法相继被提出。但是目前的研究多倾向于以需求和覆盖率作为排序准则,并且是一种静态排序。为此,提出一种基于历史信息的测试用例优先级技术,并在测试用例的执行过程中动态自适应地调整测试用例的优先级,以尽可能早地发现缺陷,达到预期的检错目标。在课题组开发的项目中运用该方法,验证了该方法的有效性。

关键词 回归测试,历史信息,测试用例优先级,自适应
中图法分类号 TP311.52 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.9.030

Self-adaptive Test Case Prioritization Based on History Information

CHANG Long-hui[1,2] MIAO Huai-kou[1,2] XIAO Lei[1]
(School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China)[1]
(Shanghai Key Laboratory of Computer Software Testing & Evaluating, Shanghai 201112, China)[2]

Abstract Test case prioritization(TCP), which can effectively improve the testing efficiency and reduce testing time overhead and labor costs in iterative software development process, has attracted widespread attention of researchers. And many optimization methods have been proposed. But most methods incline to TCP technology based on requirement and coverage, and keep a static sort. This paper presented a TCP technology based on history information, and we dynamically adjusted the prioritization of test cases during the execution of test cases. This method helps to find defects as early as possible and to achieve the goal of bug detection. Finally, we applied our method to the project developed by our research group to verify the effectiveness of our method.

Keywords Regression test, History information, Test case prioritization, Self-adaptive

1 引言

回归测试作为软件生命周期的一个重要组成部分,渗透在软件开发的各个阶段,在整个软件开发过程中占有很大的工作比重[1],尤其在快速迭代开发中,新版本的连续发布使得回归测试进行得更加频繁。但是,由于执行全部的测试用例会耗费大量的资源和人力,受到时间、资源和人力的限制,在每次回归测试中执行全部测试用例变得不切实际。在此情况下,为了降低回归测试执行的成本,并保证测试用例集的缺陷检测能力,研究人员提出了各种优化方法,如测试用例优先级技术。测试用例优先级技术是根据一定的排序准则将现有测试用例进行优先级排序,检错能力或覆盖率越强的测试用例其优先级越高。优先执行优先级高的测试用例,能够保证尽早地发现软件中的缺陷或实现预期的覆盖率。同时,还可以根据优先级的排序将测试用例集分成不同的等级,在不同的开发阶段执行不同等级的测试用例集,避免每次回归测试都要执行全部测试用例。通过实验表明,测试用例优先级技术有助于测试人员尽早达到预期的覆盖率目标或检错目标,提高测试效率,减少测试时间和成本[2,3]。

在回归测试中,存在多种因素影响着测试用例的优先级排序,这些影响因素即可作为优先级的排序准则。现有研究的排序准则包括需求、覆盖率、历史信息和费用等,如 Hema Srikanth[4]等研究了基于需求的回归测试优先级技术,其中考虑了需求波动性、客户定义优先级、实现复杂度和缺陷倾向性4个因素,并介绍了这4个需求值在开发流程中的获取渠道和计算方法。Elbaum[5]等分别针对函数覆盖、对重要代码覆盖和语句覆盖的情况,给出了12种不同的测试用例排序方法。Jung Min Kim[6]等采用指数平滑法对回归测试中历史执行信息进行分析,并对测试用例集进行排序。Bo Qu[7]等提出了一种算法,即根据用例发现缺陷的情况将测试用例集排序,并且提出了一种基于测试用例设计信息的优先级算法[8]。Walcott[9]等提出了按照测试用例执行时间的耗费对其进行优先级排序的方法。在回归测试执行中,这些排序准则既可以单独执行也可以相互结合执行,测试者可以根据当前测试的需求赋予排序准则不同的权重。

历史信息是后继回归测试优先级执行的重要排序准则,其记录的执行信息(如测试用例的缺陷检测情况和用例执行的稳定性等)既反映了系统的修改变化,也暗示了当前回归测

到稿日期:2014-10-27 返修日期:2014-12-21 本文受国家自然科学基金项目(61170044,60970007),上海市教委科研项目(13YZ141)资助。
常龙辉(1988-),女,硕士生,主要研究方向为基于模型的测试用例优化生成,E-mail:changlhzjut@163.com;缪淮扣(1953-),男,教授,博士生导师,主要研究方向为软件形式化方法、软件测试;肖 蕾(1979-),女,博士生,副教授,主要研究方向为软件测试。

试的重点。本文提出了一种基于历史信息的测试用例优先级方法，根据历史信息中用例发现缺陷的情况和用例执行的稳定性对测试用例集排序，并在回归测试执行中动态调整优先级。该方法有助于在有限的时间内及早发现软件中更多的缺陷，提高测试用例集的检错效率。

2 优先级排序的相关概念

测试用例优先级是指根据一定的排序准则将测试用例集进行排序，使得满足测试要求的测试用例较先被执行[10]。由于不同的测试用例具有不同的错误检测能力和代码或需求的覆盖能力，因此若将测试用例按照一定的准则排序，提高检错能力或覆盖能力较强的测试用例的优先级，有助于在较短的时间内达到预期的缺陷检测率和代码或需求覆盖率。对于测试用例优先级设定的问题，Rothermel[11]等将其形式化地描述为：给定测试用例集 T，T 中测试用例的所有可能排序的集合 PT，以及排序目标函数 f。f 的定义域为 PT，值域为实数。测试用例排序的目的是找出 $T''\in PT$，并且满足：

$$(\forall T'')(T''\in PT)(T''\neq T')[f(T')\geqslant f(T'')]$$

其中，f 是对排序目标的定量描述，用于度量排序的有效性，f 值越大，该测试用例集排序效果越好，其检错效率或覆盖能力越高。

为了能够形式化地说明优化的测试用例序列检测出缺陷的速率，Rothermel 等提出了缺陷检测加权百分比（Average of the Percentage of Faults Detected，APFD）[10,12]的软件测评指标。在执行排序后的测试用例时，由 $APFD$ 可以得到在执行测试用例过程中检测缺陷的平均累计比例。其公式描述如下：

$$APFD=1-\frac{TF_1+TF_2+\cdots+TF_m}{nm}+\frac{1}{2n}$$

其中，n 表示测试用例集 T 中共有 n 个测试用例，m 表示该测试用例集可检测出 m 个缺陷，TF_i 表示在排序后的测试用例集 T' 中第一个发现缺陷 i 的测试用例在该执行次序中所处的次序。$APFD$ 是一个百分数，其值越大，表明该序列的测试用例集的缺陷检测速度越快。

3 优先级排序

在迭代式软件开发流程中，需要针对每个开发版本进行测试，而已执行的回归测试的结果影响着后继回归测试的执行，也即回归测试的历史信息对后续的测试执行有一定的参考价值，如回归测试历史信息中用例发现缺陷的情况、测试用例执行的稳定性、测试用例的执行时间等。在工程实践环境中，由于只能有部分信息便于从历史信息中得到，而发现缺陷个数和用例执行的稳定性可以较直观地从执行结果中获取，且对后继回归测试有重要影响，因此本文根据这两个排序准则对测试用例进行优先级排序，并且在测试用例执行的过程中动态调整测试用例的优先级。

3.1 执行过程

在动态监控测试用例的执行过程中，需要先依据历史信息初始化测试用例优先级，然后随着用例的逐个执行，根据当前的执行结果以及测试用例的关联性，动态调整优先级序列，本文根据功能域动态调整优先级。功能域是根据待测系统对所有测试用例进行功能性划分，不同功能域间用例不存在关联性，相同功能域内用例存在关联性。如对于一个简单的邮件系统，可以将其划分成以下 4 个功能域：登录、写信、收信、草稿箱。同时，由于根据功能域动态调整测试用例的优先级，功能域分组的粒度不宜过大，否则会导致优先级大幅度的调整，影响排序效果。

当一个测试用例发现缺陷时，根据软件程序的相关性，处于同一功能域的用例可能发现与已发现的缺陷相关的缺陷。为此，提高与已发现缺陷的用例处于同一功能域的测试用例的优先级，从而集中对已发现缺陷的功能，进行测试，以获得较高的缺陷检测率。基于历史信息的测试用例优先级动态调整执行过程描述如下：

1. 依据测试用例所覆盖的功能，将所有的测试用例划分功能域。

2. 若是首次测试，依据用户需求或用例覆盖度排序执行；若是回归测试，根据历史信息中缺陷发现率和执行稳定性排序执行。

3. 执行排序后的首个测试用例，若检测到缺陷，则动态调整与该测试用例处于同一功能域的用例的优先级；并从测试用例集中删除当前已执行的测试用例。

4. 重复执行第 3 步，直至执行完所有的测试用例。

在上述回归测试执行步骤中，第 2 步和第 3 步最为重要。第 2 步是测试用例优先级初始化排序计算方法，3.2 节会对这一方法进行详细阐述。第 3 步为自适应优先级策略，在3.3 节通过算法介绍自动调整方法。

3.2 测试用例优先级初始化排序计算方法

在依据历史信息对测试用例初始化排序时，考虑缺陷发现情况和用例执行稳定性两个因素。根据指数平滑法[6]，最近发生的情况在某种程度上会影响到不久的将来。所以已执行的回归测试中发现缺陷或执行不稳定的测试用例在当前或后续回归测试中发现缺陷或执行不稳定的概率较大，故应提高其优先级。

(1)缺陷发现率(Bug Found Rate，BFR)。若某一个测试用例在上一次回归测试或以往的回归测试中发现缺陷，那么在下一次或未来的某个版本中会加入开发人员修复缺陷的更新代码，故在后续的回归测试中提高该测试用例的优先级既能及时确定已发现缺陷是否被修复，又可以测试更新的代码对当前系统是否植入新的缺陷。同时，在以往的回归测试中，若某个测试用例发现的缺陷的次数较多，则表明该测试用例所测试的系统功能不稳定或修改频率较大，需要进一步测试。因此，在每次回归测试执行完毕后，记录每个测试用例发现缺陷的情况，在下一次回归测试执行前根据以下公式计算当前测试用例的 BFR 值：

$$BFR_{i,j}=\frac{BF_{i,j}}{\sum_{i=0}^{i=n}BF_{i,j}}$$

其中，n 表示测试用例集中测试用例的个数；$BF_{i,j}$ 表示第 i 个测试用例在近 j 次回归测试中发现缺陷的总次数；$BFR_{i,j}$ 表示在近 j 次回归测试中，测试用例 i 发现缺陷次数占所有测试用例发现缺陷次数的比重，以此来表示测试用例 i 发现缺陷的能力。其中 j 可以根据软件开发过程中版本发布的阶段设置，若系统对某个版本已经稳定，则之后的回归测试都以这个版本为基准，而 j 就是从稳定版本到当前版本的回归测试的次数。$BFR_{i,j}$ 值越大，表明测试用例 i 发现缺陷的能力越

强，其优先级越高。

(2)测试用例稳定性(Case Stable Rate，CSR)。在回归测试过程中，若设计的测试用例无法顺利执行下去，则可能存在两个问题：1)待测系统有改动或系统测试环境不稳定；2)测试用例或测试数据设计不合理。这两种情况反映了系统或用例存在风险，需要通过重现用例的执行，确定用例无法顺利执行的原因，然后修改用例或系统。因此，在后续回归测试中，应提高该测试用例的优先级，用于检测修改后的测试用例执行的稳定性或修改后系统的正确性。依据稳定性计算测试用例优先级的公式为：

$$CSR_{i,j}=\frac{CS_{i,j}}{\sum_{i=0}^{i=n}CS_{i,j}}$$

同样，n 代表测试用例集中共有 n 个测试用例，$CS_{i,j}$ 表示第 i 个测试用例在近 j 次回归测试执行过程中出现不稳定的次数，$CSR_{i,j}$ 表示在近 j 次回归测试用中第 i 个测试用例不稳定次数占所有测试用例不稳定执行次数的比重。$CSR_{i,j}$ 值越大，表明测试用例 i 或该用例测试的功能越不稳定，其优先级越高。

(3)优先级计算。测试人员在执行回归测试时，可以根据实际需求对 *BFR* 和 *CSR* 设置不同的权重(Factor Weight，FW)，然后综合考虑这两个历史信息因素对测试用例进行优先级排序。初始情况下，为 *BFR* 和 *CSR* 设置相同的权重，则基于历史的测试用例优先级值(History Factor Value，HFV)的计算公式如下：

$$HFV_{i,j}=BFR_{i,j}*FW_1+CSR_{i,j}*FW_2$$

其中，FW_1 和 FW_2 分别代表 *BFR* 和 *CSR* 的权重，初始默认值均为 0.5。测试人员在实际操作中可以根据测试需求调整 FW_1 和 FW_2，若想要检测出更多潜在的缺陷或已发现缺陷的修改情况，可适当提高 FW_1 的值；若倾向于检测系统或用例的稳定性，可适当提高 FW_2 的值。$HFV_{i,j}$ 即是近 j 次回归测试中用例 i 的基于历史信息的优先级值，值越大，优先级越高。根据该值对测试用例初始化排序，进行当前的回归测试。

3.3 优先级动态调整

在测试用例执行过程中，若某个用例发现系统缺陷，表明其测试的功能可能存在风险，则与该功能相关的执行步骤也可能会发现相关缺陷。为了能够集中对该功能进行更全面地测试，根据测试用例划分的功能域动态调整当前测试用例的优先级。与基于历史信息的测试用例优先级初始化方法相结合，动态调整优先级算法如下：

```
Input:TC[](输入测试用例集),n(测试用例的个数)
Output:发现缺陷的测试用例
for i=0 to n
    HFV(TC[i]);//计算优先级值
RTC[]=Rank(HFV);//根据 HFV 值进行降序排列
for i=0 to n
    Excute(RTC[i]);//执行当前的测试用例
    if(RTC[i] found bug)
        Output RTC[i];
        for j=i to n
            if(RTC[j]. function==RTC[i]. function)//若后续测试用例与当前测试用例处于同一功能域,则将其优先级提高 j-i-range 个等级
                RTC[j]. priority-=(i+range);
                Update(priority);//更新所有剩余用例的优先级
            End if
        End for
    End if
End for
```

在该算法中，先根据 *BFR* 和 *CSR* 对测试用例进行优先级初始化排序，在执行排序后的测试用例的过程中，若有测试用例发现缺陷，则将与该测试用例处于同一功能域的测试用例的优先级提高 $j-i-rang$ 个等级，即将与发现缺陷的用例处于同一功能域的用例提升到当前用例后 *range* 个位置执行，以便更加集中地对已发现缺陷的功能进行测试。其中，*range* 值与测试用例的总数、功能域划分的粒度、已执行的测试用例总数和尚未执行的测试用例总数相关。就目前而言，尚不能给出精确的计算定义，可根据实践经验获得最优值。

4 实验分析

4.1 测试系统

睦邻在线(CPMISS)是我们课题组开发的提供社区物业管理及服务的信息化平台，用于为社区物业的所有人、使用人、服务和管理等各方人员提供经济的信息秘书服务。该平台主要包括业户中心、业户信函、社区楼宇和社区会议等模块，其中社区会议是我们近期开发的功能模块，该模块简单的流程如图 1 所示。该模块主要包括了会议注册、会议审核、议程互动和会议纪要 4 部分。注册的会议由若干议程组成，会议主持人可以根据需要选择会议议程类型，其中，议程类型包括图文类、选举类、表决类、发言类和讨论类。会议主持人通过新增会议确定会议主题、参会人员和会议议程，并提交审核。会议审核通过后，会议将在指定时间召开。在会议的进行中，社区参会人员可以通过会议邮件参与议程互动，如投票、表决，以及参与讨论，最后形成会议纪要。

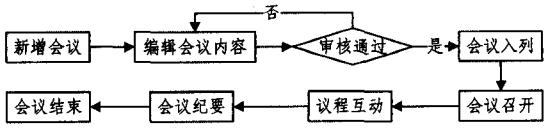

图 1 CPMISS 社区会议模块简单流程

4.2 实验及结果分析

在会议模块大量的开发工作结束后，制定了 88 条相关的测试用例，在 Firefox 浏览器 31.0 版本上进行功能测试。首先，根据测试功能为所有的测试用例划分功能域，其中包括会议注册、会议审核、图文类议程编辑、图文类议程互动、图文类会议纪要、讨论类议程编辑等。然后，在回归测试中根据历史信息对测试用例集进行排序，并在回归测试实验过程中动态调整测试用例的优先级。同时，在实验过程中，将本文提出的基于历史信息(BFR&CSR)的测试用例自适应的优先级技术与另外 3 种方法进行对比实验，对比实验方法包括：单独以缺陷发现情况(BFR)、用例稳定性(CSR)为排序准则的方法以及未优化(Unoptimized)随机排序的方法，这 3 种方法同样采用动态调整优先级的算法。在本实验中，设定一系列 *range* 值来进行比较实验，分析实验结果可知，当 *range* 值设定为 2 时，能够达到整体最好的实验效果。这是因为前后用例之间存在关联性，即当前测试用例发现缺陷时，下一个测试用例发现缺陷的概率较大，若将 *range* 值缩减为 1 就破坏了原本的

关联性优先级；若 *range* 值过大，就不能体现快速提高相同功能域优先级的特性。将 *range* 值设置为 2，表示当一个测试用例发现缺陷后，与该测试用例处于同一功能域的测试用例将会被提高到当前测试用例后的第 2 个位置执行，若有多个相关的测试用例，则在提升优先级后按原顺序排序。本文共进行了 4 轮测试，每轮测试中发现的缺陷均为真实缺陷，开发人员根据发现的缺陷修改系统，测试人员再进行下一轮的测试。这 4 轮回归测试中，首次测试中用例的优先级排序由客户需求决定，后续 3 轮回归测试根据不同准则对测试用例进行排序。本文针对比较实验的第二次和第三次回归测试实验结果进行分析。

测试用例优先级排序的目的是使得优先级高的测试用例检测到更多的缺陷。基于此，将第二次和第三次回归测试中 4 种优先级方法发现缺陷的情况汇总，如图 2 和图 3 所示。

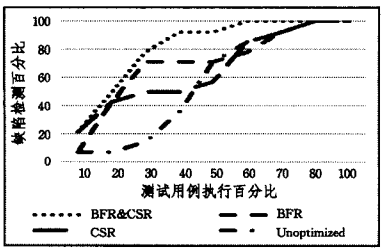

图 2　第二次回归测试各优先级方法比较

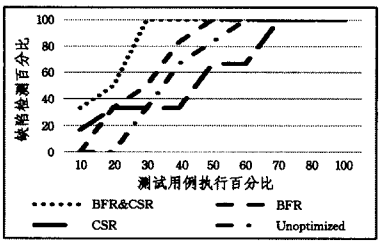

图 3　第三次回归测试各优先级方法比较

图 2 中，在执行了前 40%的测试用例时，4 种优先级方法（BFR&CSR、BFR、CSR 和 Unoptimized）发现缺陷的百分比分别是：92.3%，71%，49.7%和 35.5%，BFR&CSR 方法在相同时间内发现缺陷的数量明显高于其他方法。在这 4 轮回归测试中，第二次回归测试共发现 14 个缺陷，随机未优化方法排序（Unoptimized）和采用本文方法排序（BFR&CSR）的测试用例集的 *APFD* 值分别为 50.7%和 62.3%；第三次回归测试共发现 6 个缺陷，按随机排序和采用本文方法排序的测试用例集的 *APFD* 值分别为 56.8%和 76.5%。通过 *APFD* 值可说明本文方法能更早发现缺陷。同时由于本文方法是对历史信息中发现缺陷或不稳定次数的累加，因此在执行了两轮回归测试后，频繁检测出缺陷或不稳定的用例随着方法的迭代其优先级逐渐升高。如图 3 在第三次回归测试中，本文方法的优势更加突出，仅在执行了前 30%的测试用例后将所有的缺陷检测出来。

根据系统功能的相关性，在采用动态调整策略后，缺陷发现的位置比较集中。如图 3 所示，在执行 10%的测试用例后发现了 33.3%的缺陷，在执行 20%到 30%的测试用例时就将剩余的缺陷全部检测出来，说明当缺陷依据功能分布比较集中时，若发现其中一部分缺陷，则其他的缺陷将很快被检测出来。表 1 统计了第三次回归测试中部分功能域分组中测试用例发现缺陷的首末位置情况，以及第一次测试中发现缺陷的首末位置。从表 1 统计中可以看出，第三次 BFR&CSR 回归测试中首次和最后发现缺陷的平均位置差明显比未优化的方法小得多，表明本文方法能够快速聚拢检测相似缺陷的测试用例，并集中检测已发现缺陷的功能。同时，与第一次测试中首次发现缺陷的位置相比，BFR&CSR 方法提高了首次发现缺陷的位置，帮助测试人员尽早地发现缺陷。如针对图文类议程编辑功能域的用例，在第一次测试中首次发现缺陷的位置是 56，在第三次 BFR&CSR 回归测试中首次发现缺陷的位置是 22，首次发现缺陷的位置提高了 34。但与此同时，由于采用了动态调整的策略，会自动提升一些测试用例的优先级，但是有些测试用例是不能检测到缺陷的，这就导致那些原本优先级较高（发现缺陷概率较大）的测试用例被延后执行。如图 2 中，在执行 40%到 50%的测试用例时是没有发现缺陷的，其原因是提高了与发言类议程编辑相关的测试用例的优先级，而提高的测试用例中部分用例是没有检测到缺陷的。

表 1　功能域分组中用例发现缺陷的位置

功能域	第一次测试		第三次测试 Unoptimized		第三次测试 BFR&CSR	
	首次发现缺陷的用例的位置	最后发现缺陷的用例的位置	首次发现缺陷的用例的位置	最后发现缺陷的用例的位置	首次发现缺陷的用例的位置	最后发现缺陷的用例的位置
会议审核	15	29	27	27	13	13
图文类议程编辑	56	60	56	60	22	24
图文类议程现场	31	40	34	74	6	8
图文类议程会议纪要	39	45	39	45	1	1
平均位置差	8.25		12.5		1	

结束语　回归测试在整个软件维护成本中占了较大的比重。为此，本文提出了基于历史信息的测试用例优先级排序方法，其中涵盖了缺陷发现情况和用例稳定性两种历史信息，并提出了在测试用例执行过程中动态调整优先级的算法，最大程度地实现回归测试的执行效率，降低回归测试的成本。同时，以课题组开发的应用项目为例，在回归测试中实践本文方法，实验表明该方法有效提高了测试用例在回归测试中的检错能力，并能够对有风险的功能进行集中测试。本文方法既可以单独作为回归测试中优先级排序的准则，又可以结合其他的排序准则（如需求）共同作为排序准则。但是，由于本文方法是基于历史信息的，在首次回归测试中需要依赖于其他的排序准则，在实验中，首次测试的优先级排序依赖于客户需求。动态调整优先级的算法是本文方法实现的关键，该算法中 *range* 值的设定对执行效果起关键作用，而 *range* 值与测试用例的总数、功能域划分的粒度、已执行的测试用例总数和尚未执行的测试用例总数相关。就目前而言，该值的设定倾向于经验值，其更精确的计算方法是以后讨论并解决的问题。

参 考 文 献

[1] Rothermel G, Harrold M J. Analyzing regression test selection techniques [J]. IEEE Transactions on Software Engineering, 1996, 22(8): 529-551

[2] Rothermel G, Untch R H, Chu C Y, et al. Prioritizing test cases for regression testing [J]. IEEE Transactions on Software Engineering, 2001, 27(10): 929-948

[3] Li Z, Harman M, Hierons R M. Search algorithms for regression test case prioritization [J]. IEEE Transactions on Software Engineering, 2007, 33(4): 225-237

[4] Srikanth H, Williams L, Osborne J. System test case prioritization of new and regression test cases[C] // 2005 International Symposium onEmpirical Software Engineering, 2005. IEEE, 2005: 10

[5] Elbaum S, Malishevsky A G, Rothermel G. Prioritizing testcases for regression testing [C] // Proceedings of the International Symposium on Software Testing and Analysis. 2000: 102-112

[6] Kim J M, Porter A. A history-based test prioritization technique for regression testing in resource constrained environments[C] // Proceedings of the 24rd International Conference on Software Engineering(ICSE 2002). IEEE, 2002: 119-129

[7] Qu Bo, Nie Chang-hai, Xu Bao-wen, et al. Test case prioritization for black box testing[C] // 31st Annual International Computer Software and Applications Conference, 2007(COMPSAC 2007). IEEE, 2007, 1: 465-474

[8] 屈波，聂长海，徐宝文. 基于测试用例设计信息的回归测试优先级算法[J]. 计算机学报，2008，31(3)：431-439
Qu Bo, Nie Chang-hai, Xu Bao-wen. Test case prioritization based on test suite design information [J]. Chinese Journal of Computers, 2008, 31(3): 431-439

[9] Walcott K R, Soffa M L, Kapfhammer G M, et al. Timeaware test suite prioritization[C] // Proceedings of the 2006 International Symposium on Software Testing and Analysis. ACM, 2006: 1-12

[10] Elbaum S, Rothermel G, Kanduri S, et al. Selecting a cost-effective test case prioritization technique[J]. Software Quality Journal, 2004, 12(3): 185-210

[11] Rothermel G, Untch R H, Chu C, et al. Prioritizing test cases for regression testing[J]. IEEE Transactions on Software Engineering, 2001, 27(10): 929-948

[12] 屈波，聂长海，徐宝文. 回归测试中测试用例优先级技术研究综述[J]. 计算机科学与探索，2009，3(3)：225-233
Qu Bo, Nie Chang-hai, Xu Bao-wen. Survey of Test Case Prioritization for Regression Testing [J]. Journal of Frontiers of Computer Science and Technology, 2009, 3(3): 225-233

(上接第 138 页)

[4] 陈波，师惠忠. 一种新型 Web 应用安全漏洞统一描述语言[J]. 小型微型计算机系统，2011，32(10)：1994-2001
Chen Bo, Shi Hui-zhong. Novel uniform vulnerability description language of Web application[J]. Journal of Chinese Computer System, 2011, 32(10): 1994-2001

[5] Jiang F, Dong Dao-yi, Cao Long-bing, et al. Agent-based self-adaptable context-aware network vulnerability assessment[J]. IEEE Transaction on Network and Service Management, 2013, 10(3): 255-270

[6] 陆余良，夏阳. 主机安全量化融合模型研究[J]. 计算机学报，2005，28(5)：914-920
Lu Yu-liang, Xia Yang. Research on target-computer secure quantitative fusion model [J]. Chinese Journal of Computers, 2005, 28(5): 914-920

[7] 周亮，李俊娥，陆天波，等. 信息系统漏洞风险定量评估模型研究[J]. 通信学报，2009，30(2)：71-76
Zhou Liang, Li Jun-e, Lu Tian-bo, et al. Research on quantitative assessment model on vulnerability risk for information system [J]. Journal of Communications, 2009, 30(2): 71-76

[8] 杨宏宇，朱丹，谢丽霞. 网络信息系统漏洞可利用性量化评估研究[J]. 清华大学学报(自然科学版)，2009，49(S2)：2157-2163
Yang Hong-yu, Zhu Dan, Xie Li-xia. Quantitative evaluation of vulnerability exploitability in network information systems[J]. Journal of Tsinghua University(Science and Technology), 2009, 49(S2): 2157-2163

[9] 宋舜宏，陆余良，杨国正，等. 一种应用主机访问图的网络漏洞评估模型[J]. 小型微型计算机系统，2011，32(3)：483-488
Song Shun-hong, Lu Yu-liang, Yang Guo-zheng, et al. Network vulnerability assessment model applying host-based access graphs [J]. Journal of Chinese Computer Systems, 2011, 32(3): 483-488

[10] 李鑫，李京春，郑雪峰，等. 一种基于层次分析法的信息系统漏洞量化评估方法[J]. 计算机科学，2012，39(7)：58-63
Li Xin, Li Jing-chun, Zheng Xue-feng, et al. Analytic hierarchy process (AHP)-based vulnerability quantitative assessment method for information systems[J]. Computer Science, 2012, 39(7): 58-63

[11] 王新喆，许榕生. 基于 CVE 漏洞库的生存性量化分析数据库和量化算法的设计[J]. 计算机应用，2008，28(2)：415-417，421
Wang Xin-zhe, Xu Rong-sheng. Design of survivability quantum analysis database and quantum algorithm based on CVE database[J]. Computer Applications, 2008, 28(2): 415-417, 421

[12] Liu Qi-xu, Zhang Yu-qing. VRSS: A new system for rating and scoring vulnerabilities[J]. Computer Communications, 2011, 34(3): 264-273

[13] Martin R A. Making security measurable and manageable[C] // Proceeding of the 2008 IEEE Military Communications Conference. San Diego, CA, 2008: 1-9

[14] Microsoft security response center security bulletin severity ratingsystem[EB/OL]. http://www.microsoft.com/technet/security/ bulletin/rating.mspx, 2012

[15] Vupen security [EB/OL]. http://www.vupen.com/english, 2012

[16] US-CERT. Vulnerability notes database field descriptions[EB/OL]. http://www.kb.cert.org/vuls/html/fieldhelp # metric, 2012

[17] IBM IIS X-Force[EB/OL]. http://xforce.iss.net, 2012

[18] China National Vulnerability Database of Information Security [DB/OL]. http://www.cnnvd.org.cn/vulnerability, 2014