浙江理工大学学报(自然科学版),第 37 卷,第 2 期,2017 年 3 月
Journal of Zhejiang Sci-Tech University (Natural Sciences)
Vol. 36, No. 2, Mar. 2017

DOI:10.3969/j.issn.1673-3851.2017.03.014

基于缺陷传递的缺陷关联系数调整策略研究

董　萌,包晓安,张　娜,熊子健,俞成海
(浙江理工大学信息学院,杭州 310018)

摘　要:在软件开发中,软件缺陷不能完全独立存在,而是存在关联关系。通过分析软件缺陷关联,可以更好的检测和剔除某种特定类型缺陷。文章从软件缺陷管理的角度出发,对缺陷关联进行分析,研究了缺陷间数据相似度、缺陷间耦合度与软件缺陷关联系数之间的关系,设计了一种基于缺陷传递的缺陷关联系数调整策略,实现对缺陷关联系数的调整。结果表明,与传统的缺陷关联系数方法和关联缺陷管理方法相比,笔者提出的策略能够在更短的时间内完成同等质量的缺陷排除任务,从而提高测试效率。

关键词:缺陷关联;缺陷管理;缺陷传播;关联系数
中图分类号:TP3　　**文献标志码**:A　　**文章编号**:1673-3851 (2017) 02-0232-05

0　引　言

随着计算机软件技术被广泛应用于各个行业,软件可靠性问题愈发被重视。尽管开发人员在开发过程中使用不同方法提高软件开发质量,但由于软件结构的日益复杂,开发人员并不能完全预防缺陷产生,软件测试仍然是提高软件质量的重要手段[1-4]。在实际测试中,人们发现软件缺陷之间无法做到完全独立存在,而是存在某种关联关系。

对于软件缺陷数据的关联关系的研究,提高了软件质量和缺陷剔除效率。分析缺陷关联系数的目的就是为了找出某个缺陷被触发时,其他缺陷被触发的可能性,作为缺陷管理和剔除的依据。利用缺陷关联系数来衡量各缺陷之间的关联关系的强弱,通过缺陷关联系数来查找和排除关联度高的一类缺陷。

本文从软件缺陷关联的原因出发,研究了缺陷间数据相似度、缺陷间耦合度对于软件缺陷关联系数的影响,建立了三者之间的关联关系。最后设计了一种基于缺陷传递的缺陷关联系数调整策略,实现对缺陷关联系数的调整。

1　相关工作

目前,绝大部分软件使用面向对象的开发方法,其继承性决定了程序块并不能完全独立,多多少少都会受到其他程序块的影响,导致程序块中存在的缺陷也存在一定的关联关系。为了对缺陷间的关联关系进行度量,研究人员需要获取和分析缺陷信息。缺陷管理理念的出现,使得人们可以对发现的缺陷进行跟踪和记录,并且缺陷管理参与到软件生命的各个周期,这使得将缺陷管理的理论引入到缺陷分析中成为一种可能[5]。

在关联缺陷领域,Katerina 等[6]首次提出失效关联的概念,认为软件某模块的失效与其他模块有关,并不是相互独立的,提出一种利用 Markov 更新模型对有失效关联的软件可靠性进行建模。景涛等[7]给出了关联缺陷的定义,利用 Space 软件[8]验证了缺陷之间的关联关系,并通过实验数据的验证提出了一种缺陷放回的测试方法来剔除关联缺陷。前期研究中,包晓安等[9]、姚弋等[10]将缺陷关联引入到 Markov 模型中,利用关联缺陷的性质将关联

收稿日期:2016－04－24　　网络出版日期:2017－01－03
基金项目:国家自然科学基金项目(61379036,61502430);浙江省重大科技专项重点工业项目 (2014C01047);浙江省自然科学基金项目(LY12F02041)
作者简介:董　萌(1989－),男,江苏徐州人,硕士研究生,主要从事软件测试方面的研究。
通信作者:张　娜,E-mail:zhangna@zstu.edu.cn

系数作为软件状态转移的向导,进而转化为带权路径问题求得缺陷关联集的最大回扣值,提高测试效率。近些年,国内外研究人员将软件缺陷管理技术应用于缺陷分类和缺陷度量,利用缺陷管理提供的数据对缺陷进行更精确的度量。Lee 等[11]提出了一种对缺陷定性的度量方法,该方法通过确认缺陷的触发原因,将相应信息反馈给开发测试人员。王斌等[12]软件缺陷分类方法是按错误性质分类,该方法不限于软件本身的错误,而是根据软件测试和使用中填写和反馈的问题报告等进行分析。李鹏等[13]则将数学相似性引入缺陷关联模型,研究了缺陷与面向对象开发之间的关系。

总的来说,人们已经认识到缺陷之间的关联关系,但对这种关系的研究才刚刚起步,没有利用缺陷管理采集到的数据对软件缺陷关联进行分析。对缺陷关联测试模型的研究中,往往对缺陷关联系数矩阵进行简单的设定和度量,并不能真实反映出实际测试环境中软件缺陷之间的关联程度,因此准确的获得关联缺陷系数矩阵就显得更重要。

2　缺陷关联系数影响因子

缺陷关联关系有 3 种不同的表现形式,既可以是基于逻辑设计的控制流缺陷,也可以是基于数据传输的数据流缺陷,或者是基于控制流和数据流相互作用的共性关联缺陷。因此,本文从数据流和控制流出发,对影响缺陷关联的两个影响因素进行度量研究,精确缺陷关联系数。

2.1　缺陷间数据相似度

在软件开发中,对象是开发的基本单元,也是缺陷产生的根本源点。每个对象都需要从别的对象获得继承或调用服务,这就使得对象之间存在一定的关联关系。如果某个对象产生缺陷,那么继承或调用该对象的其他对象产生缺陷的概率也会增加。因此对象中相同或相似数据之间的调用往往导致关联缺陷的产生。缺陷产生后,缺陷管理系统会对缺陷的一些情况进行记录,利用这些缺陷信息可以对缺陷关联进行度量。建立缺陷间的数据相似性的关系,也就建立了缺陷之间的潜在关联。因此基于缺陷管理数据库的关联分析对缺陷关联系数进行研究是可行的。

本文采用 IBM 提出的正交缺陷分类技术(orthogonal defect classification,ODC)作为缺陷数据的度量特征维度。ODC 将缺陷分为 8 种属性:缺陷的活动、缺陷的影响、缺陷触发、缺陷的修复对象、缺陷类型、缺陷限定词、缺陷来源、缺陷历史。这些属性是非二义性的,可以从不同角度来描述缺陷的特征,并且在不同的软件生命周期和产品中保持一致性,其中缺陷类型属性在缺陷度量中有重要作用。缺陷类型又分为赋值、检查、算法、时序、接口、功能、关联等二级属性。本文中选取缺陷特征集为(所属组件,缺陷类型,产生阶段,开发人员编号)。

通过对缺陷管理系统中缺陷数据缺陷类型的确定,便可以得到缺陷类型对应的特征维度值。确定了特征维度就可以用相同的特征维度对不同缺陷进行描述,建立缺陷之间的特征维度相似关系。通过聚类分析计算方法[14]对缺陷特征维度进行度量,实现缺陷数据相似度的计算。为方便计算公式理解,现给出如下定义:

定义 1　缺陷特征集。设 $\boldsymbol{A}=\{A_1,A_2,\cdots,A_n\}$ 是缺陷的特征集,对应于 ODC 技术中的缺陷属性。

定义 2　特征描述集合。设 $\boldsymbol{D}_k(i)=\{d_1,\cdots,d_j,\cdots,d_m\}$ 是缺陷 O_i 在特征 A_k 上的描述集合,d_j 是缺陷 O_i 在特征 A_k 上的一个描述元素。

定义 3　数据相似性。设 d 维模式向量 $\boldsymbol{X},\boldsymbol{Y}$ 为离散型数据,向量中每个特征值对应一种缺陷特征维度,设 x,y 分别为向量 $\boldsymbol{X},\boldsymbol{Y}$ 中某个缺陷特征维度的标识,则有:

$$\delta(x,y)=\begin{cases}0, if & x=y\\ 1, if & x\neq y\end{cases} \tag{1}$$

则两个缺陷特征维度集合间相似数据个数为:

$$d_{\mathrm{sim}}(\boldsymbol{X},\boldsymbol{Y})=\sum_{j=1}^{N}\delta(x,y) \tag{2}$$

由式(2)计算缺陷间数据相似度为:

$$d_{\mathrm{sim}}(x,y)=\frac{\sum_{j=1}^{N}[1-\delta(x,y)]}{\sum_{j=1}^{N}\delta(x,y)+\sum_{j=1}^{N}[1-\delta(x,y)]} \tag{3}$$

通过式(1)－(3)求得缺陷间数据相似度的值,建立缺陷间缺陷特征维度的关联,当发现某个缺陷时,将与该缺陷关联的缺陷按照缺陷特征相似度由高到低逐个列出,由测试人员进一步测试并排除对应的缺陷。

2.2　缺陷间耦合度

在软件开发过程中,常采用面向对象的分析和设计方法,即每个开发过程的对象要么继承上一个过程的对象,要么是在开发过程中产生的新的对象,面向对象的继承性决定了开发模型是一种层次结构。同时,面向对象的继承性也决定了缺陷传播的继承性,如果在对象开发设计中产生了缺陷,则继承或

调用该对象的类产生缺陷的概率也会很大。

组件是软体系结构的基本单位，也是缺陷度量的最小单位。组件之间通过类和类外的方法进行通信，类之间的耦合性在组件耦合性度量中扮演重要角色。通过度量组件之间的耦合度来间接得到缺陷间的耦合度是可行的。在缺陷管理中，缺陷报告会记录缺陷所在模块或组件位置，查找到相应组件所包含的所有与该组件相关的类方法集合信息。组件的类方法中包含需求类和供应类两种。需求类用关键字 require 表示，组件为了实现其功能需要调用其他组件的方法类，这些类就称为需求类。供应类则代表组件向其他组件提供类方法，使得其他组件可以调用该组件功能，用 provided 表示。

软件开发过程中，不同的开发模式的本质都是对象之间的相互转化，底层的对象需要由顶层对象转化而来，形成树状结构，底层对象视为子节点，顶层对象视为父节点。建立起树状结构中各个对象节点的关系，也就间接建立起组件关联关系。对应的，组件间也存在父子依赖关系。如果组件 x 是组件 y 的父亲节点，组件 y 对组件 x 的依赖度则应为组件 y 与组件 x 需求类的交集在组件 x 中所占比例。依赖度最大的情况是组件 y 的需求类均从组件 x 中继承或调用得到。类似的，如果组件 x 是组件 y 的孩子节点，组件 x 的所有提供类都是从组件 y 中继承而来，不从其他组件获取时，组件 x 对组件 y 依赖度达到最大。如果组件 x 和组件 y 之间不是父子关系，则需要通过其他组件的调用进行度量依赖关系。

一个系统是由一个组件集 $\boldsymbol{C}$ 和组件外的方法集 $\boldsymbol{M}$ 组成的。

定义 4 若组件 x 包含组件 y，则组件 x 是组件 y 的父组件，组件 y 是组件 x 的子组件。对于一个组件 $c \in \boldsymbol{C}$，则 $parient(c) \subset \boldsymbol{C}$，$child(c) \subset \boldsymbol{C}$。

定义 5 对于一个组件 $c(c \in \boldsymbol{C})$，组件 c 向外提供的方法集为 $MP(c)$，c 需求外界的方法集为 $MR(c)$。

定义 6 $P(x,y)$ 表示组件 y 需要其他组件进行交互的组件 x 提供方法集合。

综上，可以得到组件 x，y 之间的耦合度，公式表示为：

$$Depd(x,y)=\begin{cases}\dfrac{|MR(x)\cap MR(y)|}{|MR(x)|}, y\in child(x)\\ \dfrac{|MP(y)\cap MP(x)|}{|MP(x)|}, x\in child(y)\\ \dfrac{|P(x,y)|}{|MP(x)|}, y\notin child(x), x\notin child(y)\end{cases} \tag{4}$$

其中，$Depd(x,y)$ 表示组件 x，y 间的耦合度，MR 表示需求类，MP 表示提供类。

通过度量组件之间的这种依赖度，间接得到两个直接关联缺陷之间的耦合度。

2.3 缺陷关联系数的计算

从 2.1 和 2.2 中，分别从数据流和控制流的角度出发，得到缺陷间数据相似度和缺陷间耦合度两个影响缺陷关联系数的因子。在实际情况下，这两者不能完全独立，而是相互作用。为了更精确的度量缺陷关联系数，在计算时应同时考虑两种影响因子，数学公式表示如下：

$$DCC(x,y)=\alpha\cdot d_{\mathrm{sim}}(x,y)+\beta\cdot Depd(x,y) \tag{5}$$

其中，$DCC(x,y)$ 表示缺陷 x 与 y 之间的缺陷关联系数，α，β 分别表示缺陷间数据相似度和缺陷间耦合度所占权重，$0\leqslant\alpha\leqslant1$，$0\leqslant\beta\leqslant1$，$\alpha+\beta=1$。

实际测试环境中，权重系数根据不同情况需要不断调整以达到更好的测试结果。目前权重系数大小的选取主要依靠人工经验，在未来研究中可利用专门的权重系数选择方法确定，目前较成熟的方法 Delphi 法、AHP 法、专家评分法等。

3 缺陷关联系数传递

由于软件缺陷具有传播性，导致两个缺陷之间可能会得到不同的缺陷关联系数。如图 1 所示，缺陷 ***A*** 和缺陷 ***B*** 直接缺陷关联，只能得到一个缺陷关联值，而缺陷 ***B*** 和缺陷 ***G*** 是非直接关联缺陷，存在两条缺陷传播路径 ***B*-*D*-*G***，***B*-*E*-*G***，则会得到两个不同的缺陷关联系数。为避免非直接关联缺陷存在不同缺陷关联系数的情况，本文在两个缺陷之间存在多种缺陷关联性时，取关联度最高者，而不是加权求和的方法[15]。在度量缺陷间的关联度时，直接关联的缺陷之间的关联度取值为缺陷间最大关联度，非直接关联的缺陷间的关联度是取相邻缺陷间关联度的最小值。

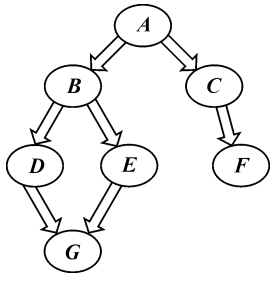

图 1 缺陷传播示意

本文所求解的是缺陷之间的关联性，根据前文分析，非直接关联缺陷的关联性是取其所有路径中

关联度最小值。本文将非直接关联缺陷间关联度问题转化为有向图最短路径计算问题,并设计了一种改进的路径搜索算法,用于求解非直接关联缺陷间的缺陷关联系数。

算法整体思想步骤如下:

步骤 1　在初始缺陷关联系数矩阵集合 $\boldsymbol{S}_{org}$ 中取除原点 V_s 外任一点作为目标点 V_{dst},定义集合 $\boldsymbol{R}_{sd}$ 为起点到终点所经过路径的权值集合,初始化为空集。

步骤 2　任取一条从 V_s 出发的路径,利用 Dijkstra 算法开始查找路径 $\forall < V_s, V_i >$,其中 $V_i \in S_{org}$,申请变量 V_{min} 用于保存当前路径最小权值,初始值为 1,然后更新走过的路径点集合 $\boldsymbol{P} = \boldsymbol{P} \cup \{V_s\}$。

步骤 3　判断有无回路:a) 若 $V_i \in \boldsymbol{P}$,存在回路,跳出循环;b) 若 $V_i \notin \boldsymbol{P}$,则转至步骤 4。

步骤 4　将当前路径值与变量 V_{min} 进行比较,取值小者。

步骤 5　判断是否已经到达终点,若到达则将 V_{min} 并入集合 $\boldsymbol{R}_{sd}$,递归返回;若未到达,则将 V_i 作为新的起点调用步骤 3,并重新申请变量 $V_{min} = 1$。

步骤 6　到达终点后,求得集合 $\boldsymbol{R}_{sd}$ 中最大的值,将其作为点 V_s 到点 V_{dst} 的缺陷关联系数权值。

步骤 7　重复步骤 1－5,直到求出 V_s 到其余所有顶点的缺陷关联系数权值。

由于本算法在计算单个定点到其他所有顶点缺陷关联系数权值具有独立性,因此可以同时重复步骤 1－7,计算出所有每两点间的关联权值计算。算法的时间复杂度为 $O(m^3)$。但笔者所提方法依赖于缺陷管理信息,在测试过程中需要收集缺陷管理信息,一定程度上导致在缺陷关联系数的计算效率不高。随着缺陷信息收集完成,可以极快的提升计算效率,将计算时间从数小时减少至几分钟。

4　实验仿真

为验证本文提出的缺陷关联系数度量技术有效性,将缺陷关联系数度量策略引入到基于缺陷关联的马尔卡尔模型中[9],将其分别与传统的缺陷关联策略以及缺陷放回策略进行对比试验[7]。在传统缺陷关联系数中,缺陷间存在关联,则缺陷关联系数定义为 1,否则为 0,缺陷关联矩阵由 0 和 1 组成。选用 Space 软件作为 benchmark 来进行测试数据比较,Space 是一个计算机研究领域公认的典型测试对象,Space 软件中存在 183 个关联缺陷对(不考虑与自身关联)[7]。用这 3 种测试策略分别进行 60 次测试,并记录前 2000 步内的实验结果。

4.1　缺陷检测率比较

在软件测试中,发现和剔除缺陷是一项基本也是极其重要的任务和目标。缺陷检测率是衡量软件质量控制的一项重要指标,它可以评估缺陷的过滤能力,评价测试策略的优劣,因而在软件的各个评审阶段中都发挥着不小的度量作用。在图 2 中,横轴表示的是测试步骤数,纵轴表示的是缺陷检测率。

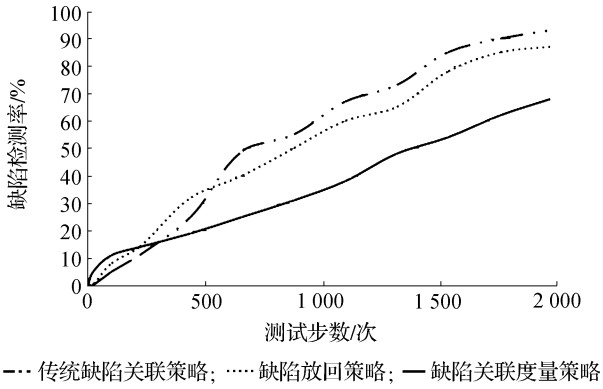

图 2　前 2000 步的软件缺陷检测率对比

由图 2 可得出,在测试初期,由于缺陷放回策略需要不停的进行一定比例的缺陷放回,本文所提测试策略需要进行关联信息的判定和计算,缺陷的检测率反而没有随机测试策略高。随着测试的持续进行,随机测试策略没有关注缺陷的关联性,因此缺陷检测率很快达到一个瓶颈期,呈缓慢增长趋势。由于缺陷放回策略和本文所提策略的检测率可以更准确的判定缺陷间的关联性,使得缺陷检测准确性提高,缺陷检测率继续上升。当缺陷检测率曲线均平稳后,在相同测试步数的条件下,放回策略缺陷检测率的平均值比传统策略高 8.48%,本文所提缺陷度量的自适应策略比缺陷放回策略高 1.84%。

4.2　缺陷回扣比较

在前期研究中,包晓安等[9]提出缺陷回扣这一概念用来描述在测试过程中检测到不同重要性等级的缺陷所产生的测试价值大小。由于缺陷有不同的类型,每个缺陷被发现的概率与剔除的难易程度都不一样,这就导致每个缺陷所产生的回扣也可能不一样。

随机选取 10 次对比实验结果进行对比分析。在图 3 中,横轴表示的是测试次数,纵轴表示的是检测所有缺陷所需。由图 3 可知,当检测完所有缺陷时,在产生的回扣方面,经过缺陷关联系数度量测试策略产生的回扣的平均值 220.88 高于缺陷放回策略的 160.24 和传统缺陷关联策略的 132.51,分别比缺陷放回策略和传统缺陷关联策略平均高 37.84%和 66.67%的缺陷回扣。

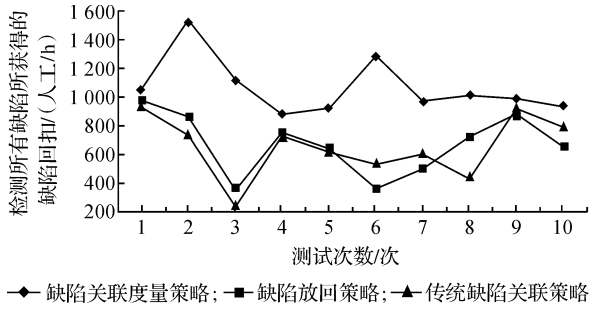

图3 第33号缺陷的缺陷检测率比较

综上可知,笔者提出的缺陷关联系数度量策略可以在相同测试条件下,更好的利用缺陷关联信息来完成测试工作,减少人力和测试资源,提高关联缺陷的检测率。

5 结 语

随着关联缺陷越来越被重视,对缺陷关联关系的研究正成为软件测试研究领域的热点。本文从软件缺陷管理的角度出发,研究了缺陷间数据相似度、缺陷间耦合度对缺陷关联系数的影响。并根据缺陷的传播过程,设计了缺陷关联系数调整算法。结果表明,笔者所提方法在缺陷检测的效率上有较好的效果。

参考文献:

[1]MEI H, HAO D, ZHANG L, et al. A static approach to prioritizing junit test cases[J]. Software Engineering, IEEE Transactions on,2012,38(6):1258-1275.

[2] SINGH P, TRIPATHI A K. Issues in testing of software with NFR[J]. International Journal of Software Engineering & Applications,2012,3(4):61.

[3] UTTING M, PRETSCHNER A, LEGEARD B. A taxonomy of model-based testing approaches [J]. Software Testing, Verification and Reliability, 2012, 22(5):297-312.

[4] SRIKANTH H, HETTIARACHCHI C, DO H. Requirements based test prioritization using risk factors: An industrial study [J]. Information and Software Technology,2016,69:71-83.

[5] DUNNE J, MCCANN P H, SLOYER J B, et al. Project modeling using iterative variable defect forecasts: U S 9,141,921[P]. 2015-9-22.

[6] KATERINA G P, TRIVEDI K S. Failure correlation in software reliability models [J]. IEEE Trans. on Reliability,2000,49(1):37-48.

[7] 景涛,江昌海,胡得斌,等. 软件关联缺陷的一种检测方法[J]. 软件学报,2005,16(1):17-18..

[8] ROTHERMEL G, UNTCH R H, CHU C, et al. Prioritizing test cases for regression testing[J]. Software Engineering IEEE Transactions on, 2000, 27(10): 929-948.

[9] 包晓安,谢晓鸣,张娜,等. 基于缺陷关联度的Markov模型软件优化测试策略[J]. 软件学报,2015,26(1):14-25.

[10] 姚弋,谢晓鸣,刘书涵,等. 考虑缺陷关联模型的软件优化测试策略[J]. 浙江理工大学学报,2016,35(1):78-83.

[11] LEE I, IYER R K. Diagnosing rediscovered software problems using symptoms[J]. IEEE Transactions on Software Engineering,2000,26(2):113-127.

[12]王斌,吴太文,胡培培. 软件缺陷分类和分析研究[J]. 计算机科学,2013,40(9):16-20.

[13]李鹏,赵逢禹. 软件缺陷关联分析与缺陷排除研究[J]. 计算机科学,2013,40(10):159-161.

[14]王骏,王士同,邓赵红. 聚类分析研究中的若干问题[J]. 控制与决策,2012,27(3):321-328.

[15]张良,张树有,刘晓健,等. 基于灰色关联与权重顺序交叉的复杂产品配置方案重构技术[J]. 计算机集成制造系统,2015,21(10):2564-2576.

The Research of Defect Correlation Coefficient Adjustment Strategy Based on Defect Propagation

DONG Meng, BAO Xiaoan, ZHANG Na, XIONG Zijian, YU Chenghai

(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: In the software development process, the software defect is not independent, but there is a correlation. A specific type of detects can be better detected and eliminated through software defect correlation analysis. From the perspective of software defect management, this paper analyzes defect correlation, and studies defect data similarity and the relationship between the coupling between defects and software defect correlation coefficient. Besides, this paper designs defect correlation coefficient adjustment strategy based on defect propagation. This strategy achieves adjustment of defect correlation coefficient. Experimental results show that, compared with traditional defect correlation coefficient strategy and defect management method, the strategy proposed in this paper can complete the defect removal task with equal quality within shorter time, and thus improve the testing efficiency.

Key words: defect association; defect management; defect propagation; correlation coefficient

(责任编辑:陈和榜)