软件缺陷关联分析与缺陷排除研究

李 鹏 赵逢禹

(上海理工大学光电信息与计算机工程学院 上海 200093)

摘 要 在软件开发过程中,软件缺陷具有传播的特性。缺陷的传播特性决定了缺陷之间并非独立存在,而存在相互关联,因此软件缺陷关联分析对于缺陷排除、软件质量保证、过程改进具有重要的意义。从软件缺陷关联的原因出发,基于面向对象的分析与设计模型,分析了软件缺陷的传播过程,研究了对象关联与软件缺陷关联之间的关系;依据缺陷的传播过程,建立了树状关联规则和特征相似关联规则;最后阐述了建立两种关联规则的步骤,开发了构建树状关联与特征相似关联的软件原型。

关键词 软件缺陷,缺陷关联,缺陷传播,缺陷移除

中图法分类号 TP311 **文献标识码** A

Research of Software Defects Associated Analysis and Software Defect Removal

LI Peng ZHAO Feng-yu

(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract Software defects have propagation characteristics in the software development process. The propagation characteristics of the software defects make defects related, which are needed to be located and removed. Software defects associated analysis is of great significance for software defects removing, quality assurance, and process improvement. In this paper, the propagation process of software defects was analyzed, and the relationship between object association and object defect association was established based on object-oriented analysis and design model. A tree association rules and characteristics similarity association rules were presented based on the propagation process of software defects. The procedures to create these association rules were given and the steps to build up tree association and the characteristics similarity association were presented.

Keywords Software defect, Defect associated, Defect propagation, Defects removing

软件缺陷不仅会造成软件质量下降,而且会导致软件项目延期、开发成本超支、后期维护费用增加、顾客满意度下降等一系列问题。因而,软件缺陷的预防与排除一直是工业界实践与学术界研究的热点。Humphrey 等人基于过程改进的思想,提出了软件过程能力成熟度模型 CMM(Capability Maturity Model),它通过提高开发过程的能力,进而提高软件产品的质量。CMM 模型以及改进的能力成熟度模型集成 CMMI(Capability Maturity Model Integration)在软件行业得到了广泛的实践与应用[1]。在软件缺陷排除与质量增长上,人们提出了软件故障树分析 FTA(Fault Tree Analysis)、软件失效模式与效应分析 FMEA(Failure Mode and Effects Analysis)、失效模式效应与危险性分析 FMECA(Failure Mode, Effects and Criticality Analysis)等缺陷分析与排除技术[1],提出了许多软件缺陷分析模型用于软件质量度量与可靠性预测。为了提高缺陷度量的精度,进而为软件缺陷分析与软件过程改进提供准确的数据,PutDam 等人提出了简单分类法[2],Thayer 等人提出了软件错误性质分类方法[4],IBM 提出了正交缺陷分类(Orthogonal Defect Classification,简称 ODC)[5]等技术。缺陷分类技术使得软件组织能够尽早发现缺陷,预防缺陷,指导软件的开发和软件过程的改进。

然而良好的软件开发过程并不能确保其软件产品的高质量,通常情况下,软件测试被认为是保证软件质量的最后一关。在软件业实践中,为了发现并排除软件中的缺陷从而保证产品的质量,投入在测试环节的人力比程序开发人力还要多。随着测试技术的发展,在单元测试、自动化测试、负载压力测试以及测试管理方面也涌现了大量优秀的软件测试工具[3,6-9]。

随着大量测试实践的应用与研究,人们发现缺陷之间并不是独立的,而是相互有着联系的。在文献[10]中,Katerina 和 Trivedi 引入了失效关联的概念,并提出了一种 Markov 更新模型对有失效关联的软件可靠性进行建模。Chen 等人认为测试回合不是相互独立的,在此基础上提出了一种二进制 Markov 过程模型,用来预测随机测试策略发现的软件失效数[11]。Bishop 等人指出可以用失效屏蔽效应来解释软件失效之间的关联关系,并可以通过适当的软件设计来杜绝失效关联[12]。软件缺陷数据的关联规则挖掘、统计分析等对提高软件质量起到了一定的帮助作用。

缺陷关联的研究对于缺陷查找有重要价值,也引起了许

到稿日期:2012-12-20 返修日期:2013-02-28

李 鹏 硕士,主要研究领域为软件工程,E-mail:wbslipengjob@126.com;赵逢禹 教授,主要研究领域为软件工程。

多研究者的关注。工程师发现一个缺陷时需要纵向排查，追溯缺陷产生的源头及其传播路径上的各模块，同时也需要横向排查与产生缺陷模块相关的其他模块。完成上述排查，工程师往往需要熟记每个模块的详细情况，以联想的方式决定需要排查的模块。由于人的记忆力是有限的，仅仅依靠大脑的联想往往会有遗漏。利用缺陷关联记载各模块之间的关联关系，工程师可通过此关联，清晰、快速地排查相关缺陷模块。本文是从缺陷关联的原因分析，首先研究缺陷在不同过程中和在同一过程中的传播规律，总结了缺陷的扩散传播方式。其次基于对象的概念，针对不同过程间对象的转化关系建立树状关联，又采取设定对象特征方法建立特征相似关联。利用这两种关联规则分析可能出现缺陷的对象模块。

1 缺陷传播分析

1.1 过程中缺陷传播

从工程的角度看，软件开发过程包含4个子过程，它们是需求过程、设计过程、编码过程、测试过程。每个过程都接受一系列输入，并产生一系列输出结果。过程是由开发人员、开发环境与基础设施、管理与规章制度等构成的系统。在每个过程的运行中，过程会受到各种随机因素的干扰，从而使过程的输出结果(即软件产品)不可避免地引入缺陷。这些缺陷有3个特点：1)隐蔽性，这些缺陷一般很难通过简单的审查与测试发现；2)缺陷具有继承与传递性，它不会仅停留在源头，而且会沿着各阶段对象的关联从上游阶段传播到下游阶段，从一个对象传递到另一个对象中去；3)残留性，每个阶段产生的缺陷，尽管根据不同过程的特点分别采取了复审、检测、走查、测试等手段，但仍然很难全部去除，并且会以某种方式传递到下游中去。例如，对需求阶段的错误分析，会导致设计阶段的设计错误；设计阶段的逻辑错误，会导致编码阶段的逻辑混乱等等。

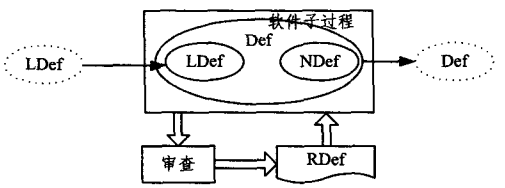

图1 对象关系示意

图1给出了软件过程的缺陷产生、移除及其传递的原理。图中各符号含义如下：

$LDef=\{ldef_1,ldef_2,\cdots,ldef_n\}$为上个过程遗漏在产品中并传递到本过程的缺陷集合，$ldef_i(1\leqslant i\leqslant n)$是上个过程传递到本过程的第$i$个缺陷。

$NDef=\{ndef_1,ndef_2,\cdots,ndef_m\}$为本过程新引入的缺陷集合，$ndef_i(1\leqslant i\leqslant m)$是本过程新产生的缺陷。

$RDef=\{rdef_1,rdef_2,\cdots,rdef_k\}$为经过审查、复查等缺陷查找方法发现并移除的缺陷集合，$rdef_i(1\leqslant i\leqslant k)$是已经移除的缺陷。

$Def=\{def_1,def_2,\cdots,def_p\}$为本过程最后所遗留的缺陷集合。

在一个过程中，各缺陷集合的关系为$Def=LDef+NDef-RDef$，其中$k\leqslant n+m$(移除的缺陷不会超过总共存在的缺陷)。本过程所遗留的缺陷Def会继续传递到下一个过程，并影响着下一个过程。因而，为了提高软件产品的质量，需要做到以下3点：1)尽量减少本过程的新引入缺陷$NDef$，增加排除缺陷$RDef$，使得遗留缺陷Def尽可能少，避免大量缺陷传递到下个过程；2)尽量追溯缺陷的源头，从源头上移除缺陷，避免缺陷继续传播；3)需要分析源头缺陷传播的范围，在范围内排查传播的缺陷。

1.2 面向对象开发的缺陷传播

在软件开发过程中，通常采用面向对象分析(OOA)与设计(OOD)技术。在整个软件过程模型中，对象是每个开发过程的基本单元，贯穿于各个阶段，缺陷也就存在于对象之中，而且会顺着对象的分析、设计与实现之间的关系传播。

图2是对象之间的关系示意。一个对象要么是继承于上个过程的某个对象(分析阶段或设计阶段的对象)，要么是过程中新产生的对象。缺陷可能来自于其继承对象的缺陷，也可能是在对象开发设计时引入的缺陷，而且缺陷会继续传递到下个过程。在同一开发或设计过程内，对象之间也具有相似性，如开发人员的知识与思维限制，一种错误会在不同的对象设计时重复出现，或者一些对象的功能、算法类似，在设计开发新对象过程中就会借鉴与其相类似的对象的算法，又或者软件开发者常用copy/paste形式开发代码。这样对象与对象之间就会存在着类似特征，即一旦对象有缺陷存在，也可能会在其他对象之中存在。

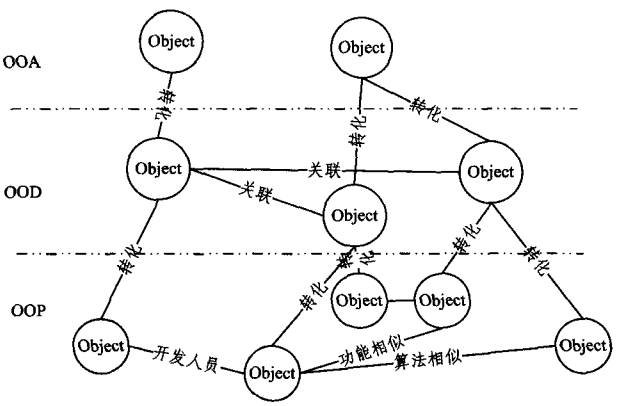

图2 对象关系示意

在一个庞大的软件系统中，上述两种缺陷传播途径一般会同时存在，缺陷的产生错综复杂，所以缺陷的传播方向是不易跟踪定位的。因而需要梳理好缺陷之间的关系并用具体的形式展现出来，才能让人们清楚地了解系统中的缺陷传播分布。依据对象间的关系来建立对象间的关联图，便于实现软件缺陷的快速跟踪定位，对缺陷查找和排除有着重要意义。

2 缺陷关联与对象关联分析

2.1 缺陷关联与对象关联

在面向对象的软件开发与设计过程中，对象是每个开发过程的基本单元，存在于每个阶段。对象之间存在关联与相似性，而缺陷存在于对象中，也就具有这种关联性与相似性，如果能够获取到对象之间的关联，那么缺陷之间的潜在关联也就很明确了，因此基于对象的关联分析是研究缺陷关联分析的可行方案。本文主要站在对象之间关联的角度研究缺陷

关联，将 OOA、OOD、OOP 中的对象关联起来，确定软件开发生命周期每个过程对象之间的关联性，保证系统从需求分析、设计、编码、测试到维护阶段，能够从一个对象缺陷出发追溯并定位关联缺陷的产生点，并基于对象间的关联关系查找缺陷。

本文将对象关联分为两类，一是树状关联，二是特征相似关联。

2.2 树状关联

树状关联是指面向对象分析、设计、实施等不同过程阶段所产生的对象之间的联系。由于每个过程的对象都是根据上一个过程的对象转化过来的，因此对象之间必然存在着明显的关联关系。

如图 2 所示，OOD 的对象模型由 OOA 的对象模型转化而来；OOP 的对象模型由 OOD 的对象模型转化而来。将每个对象模型的相互转化关系构建出来，自上而下便形成成了一种树状关联，树中的每个子节点对象都依据其父节点对象转化（或者称为具体化）而来，将树状结构中每个节点以及节点之间的关联管理起来，便可通过分析树状结构追踪缺陷并定位可能存在缺陷的其他对象。

如果检测到 OOP 阶段中一个对象内存在一个逻辑缺陷，而缺陷引入原因为其对象上层的父节点 OOD 阶段的对象设计错误造成的，那么此父节点的其他子节点有同样逻辑缺陷的可能性就会很高。使用这种树状关联，通过树形结构去查找相关待查对象，能提高缺陷排除的效率与准确性。

2.3 特征相似关联

特征相似关联是用不同特征维度描述一个对象，不同对象之间通过特征维度描述集合的相似度建立关联。一旦对象之间建立了关联，对象中的缺陷也就潜在地关联了起来，因此缺陷也可通过对象的特征相似关联的方法跟踪定位。

为了便于缺陷的跟踪定位，在选取对象特征维度时，本文参考了 IBM 提出的 ODC（正交缺陷分类）技术[5]。ODC 将缺陷分为 8 种属性（活动、触发、影响、修复对象、缺陷类型、限定词、历史、来源），这些属性两两正交，分别从不同方面描述了缺陷的特征，而且具有软件生命周期内一致性和产品间一致性等特点。其中缺陷类型属性在缺陷的定位分析和排除分析上尤为重要，缺陷类型又分为功能、赋值、接口/时序、用户接口/检查、算法等子属性。本文抽取开发人员特征、开发时间特征和缺陷类型子属性（功能、算法）作为对象的特征维度，表 1 给出了对象的特征维度以及每个维度的描述范围和作用，这些特征维度与缺陷类型对应。确定了缺陷类型，便可锁定缺陷所在对象的特征维度，通过一定的计算方法计算出该对象与其他对象间该特征维度的相似值并基于该相似值建立关联，这样就定位出存在的类型缺陷对象。例如，当定性软件缺陷的缺陷类型为功能缺陷时，则可通过建立对象在功能这个特征维度上的相似关联，来排查相关对象是否有类似缺陷存在。

对象特征维度建立后，通过集合计算方法来实现对象特征相似度的计算。为了便于对象的特征相似度计算的理解，首先给出以下几个定义。

定义 1（对象特征集） 设 $A=\{A_1,A_2,\cdots,A_n\}$ 是对象的特征集，表 1 给出了一般的对象特征及其所描述的范围。

表 1 相似关联的对象特征

特征	提取方法	度量内容	计算方法
复杂度	信息隐含在代码之中，提取需要分析代码。	圈复杂度、循环嵌套深度	主要分析分支、循环等语句的复杂情况，比较两个类中方法的圈复杂度。
数据结构		数据类型	抽取特征描述集合，计算集合相似的值作为特征相似值。
继承关系		继承的父类、接口	
功能	信息存在于代码注释中，提取注释中的信息。	实现的功能	
人员		开发人员，设计人员	
时间		设计时间，编码时间	

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

定义 3（特征相似值） 设 $R(O_i,O_j,A_k)$ 是对象 O_i 与 O_j 在 A_k 特征上的相似值，则

$$R(O_i,O_j,A_k)=\frac{2|D_k(i)\cap D(j)|}{|D_k(i)|+|D_k(j)|} \tag{1}$$

式中，R 的值域为 $[0,1]$，$|D_k(i)|+|D_k(j)|$ 表示 $D_k(i)$ 集合与 $D_k(j)$ 集合元素个数的和。

利用式(1)，我们可以计算出数据结构、继承关系、功能、人员、时间特征上的相似程度。例如 A、B 两个对象中的数据结构集合分别是(int,string,list,class1,class2,class3)和(int,string,class1,Dictionary,class4,class5)，那么它们的数据结构特征的特征交集为(int,string,class1)，所以特征相似度为 $\frac{2\times3}{6+6}=0.5$。

通过上述计算方法计算出相似度后，建立对象的相似关联，当发现缺陷并确定其缺陷类型时，将与该缺陷类型一致的对象按照特征相似度由高到低逐个列出，由测试人员进一步测试在相似的对象中是否存在缺陷。

3 建立关联的技术步骤

实现关联分析的关键就是针对树状关联和特征相似关联分别建立关联树和相似矩阵，下面阐述建立两种对象关联的步骤。

3.1 树状关联的构建

在面向对象的分析与设计中，对象是用类来描述的，对象之间的关系体现为对象所对应的类间的关系。为了建立对象的树状关联，需要对软件分析、设计、编码过程建立的文档进行分析，抽取文档中所有类，找出不同阶段中类的关联关系。构建类的树状关联主要步骤如下：

1）在面向对象分析、设计、实施文档中，搜索过程文档中的类，将这些类记录下来。

2）分析类之间的转化（细化）关系，建立当前文档中的类与开发上游过程的文档中类之间的转化关系。

3）将分析结果转化为关联树，确定该类与上个过程中哪个类相关，以确立其父节点。存储关联树结构，只需将每个类设置父节点字段表明其父节点对象，若无父节点可为空。

3.2 特征相似关联的方法

为了便于分析，本文把特征相似关联的研究集中于程序代码级对象相似关联。实际上，对开发设计的每个阶段，都可

（下转第 189 页）

[32] He Hao, Wang Hai-xun, Yang Jun, et al. BLINKS: Ranked Keyword Searches on Graphs[C]// ACM Int. Conf. on Management of Data(SIGMOD). Beijing, China, 2007: 305-316

[33] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases[C] // 18th Int. Conf. on Data Engineering(ICDE). 2002: 5-16

[34] Hristidis V, Gravano L, Papakonstantantinou Y. DISCOVER: Keyword Search in Relational Databases[C]// 28th Int. Conf. on Very Large Data Bases(VLDB). Hong Kong, China, 2002: 670-681

[35] Balmin A, Hristidis V, Papakonstantinou Y. ObjectRank: Authority-Based Keyword Search in Databases[C] // 30th Int. Conf. on Very Large Data Bases(VLDB). Toronto, Canada, 2004: 564-575

[36] 文继军，王珊. SEEKER：基于关键词的关系数据库信息检索[J]. 软件学报，2005，16(7)：1270-1281

[37] 王佳宜，杨路明，谢东，等. 关系数据库上关键词的数字属性模糊查询研究[M]. 计算机技术与应用进展，2007：650-655

[38] 杨路明，王佳宜，谢东. 关系数据库上基于非数值属性关键词的模糊查询[J]. 计算机科学，2008，35(6)：236-239

[39] 金宗安，杨路明，谢东. 关系数据库模糊查询的研究[J]. 计算机工程，2009，35(13)：63-64

[40] 王佳宜，杨路明，谢东，等. 基于关系数据库的关键词查找排序策略[J]. 计算机工程与技术，2008，29(10)：2566-2569

[41] 王佳宜. 基于关系数据库的关键词模糊查询及结果集排序策略研究[D]. 长沙：中南大学，2008

[42] 黎方正. 关系数据库的关键词检索技术研究[D]. 长沙：中南大学，2010

[43] 张俊，邵仁俊，曾一鸣. 对象级别的关系数据库信息检索技术研究[J]. 计算机科学，2012，39(1)：142-147

(上接第 161 页)

以建立分析与设计文档中对象的特征相似关联。对于程序代码文档，采取自动化方法提取程序代码文档中每个类的开发人员、开发时间、类与方法功能、算法复杂度与数据结构等特征信息。构建特征相似矩阵的具体步骤如下：

1)扫描程序代码文档，提取所有类及其数据结构、复杂度、功能、人员、时间信息。其中功能、人员、时间信息在代码注释中提取；获取数据结构信息可事先建立数据类型库，扫描代码和数据类型库比对找出用到的数据类型；获取复杂度信息，通过分析分支、循环语句来计算圈复杂度和嵌套深度。

2)对每个特征建立一个 $n*n$(n 为类的个数)的相似矩阵，如图 3 所示。其中 x_{ij} 表示基本类 i 与基本类 j 在一个特征维度上的相似度。第 i 行表示基本类 i 相关的所有类的特征相似情况。

$$\begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ i \\ \vdots \\ n \end{array}\begin{array}{c} \begin{array}{cccccc} 1 & 2 & \cdots & j & \cdots & n \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix} \end{array}$$

图 3　特征相似矩阵

3)比较两个类在同一特征上的特征信息，计算出两个类在这一特征维度上的相似值。其中在功能、人员、时间、数据结构特征维度上的计算，首先要对特征建立描述集合，计算描述集合的相似值即该特征的相似值。

4)将上述计算的值写入相应的矩阵中。

最终依据特征形成对应的特征相似矩阵，确定缺陷属性依据对应的特征相似矩阵，将与之相似度最高的 n 个类列出，由测试人员进一步测试是否存在缺陷。

结束语　基于缺陷传播的思想，从对象关联的角度研究缺陷关联，提高缺陷查找的效率。但是本文给出的关联规则的有效性尚需要进一步验证，对象特征集也不完全。实际上，有关软件开发过程、技术和资源的许多问题都与软件缺陷密不可分。本文已经完成了树状关联系统的初步开发，实现了对象之间树状关联的建立，依据问题对象和树状关系给出排查对象范围。特征相似关联系统完成了部分模块算法的设计和开发。在以后的研究中，逐步完成特征相似关联的系统实现，并且在实践项目中验证，收集查找缺陷的准确率与效率数据，调整排查范围方案。

参 考 文 献

[1] 陈志元，任欣，闵蓓尔. 自动测试软件可靠性量化评估技术研究[J]. 电子测试，2008，6：24-26

[2] 聂林波，刘孟仁. 软件缺陷分类的研究[J]. 计算机应用研究，2004，21(6)：84-86

[3] 王青，伍书剑，李明树. 软件缺陷预测技术研究[J]. 软件学报，2008，19(7)：1565-1581

[4] Song Q B, Shepperd M, Cartwright M, et al. Software Defect Association Mining and Defect Correction Effort Prediction[J]. IEEE Transactions on Software Engineering, 2006, 32(2): 69-82

[5] IBM Research Center for Software Engineering Orthogonal defect classification [EB/OL]. http://www.research.ibm.com/softeng/ODC/DETODC.HTM, 2002

[6] 李福川，宋晓秋. 软件测试中的新方法——区间代数方法[J]. 计算机工程与设计，2005(10)：2576-2578

[7] 肖庆，宫云战，杨朝红，等. 一种路径敏感的静态去诶按检测方法[J]. 软件学报，2010，21(2)：209-217

[8] Morisaki S, Monden A, Matsumura T. Defect Data Analysis Based on Extended Association Rule Mining[C]//Fourth International Workshop on Mining Software Repositories(MSR'07). 2007

[9] Karthik R, et al. Defect Association and Complexity Prediction by Mining Association and Clustering Rules[C] // IEEE 2010 2nd International Conference on Computer Engineering and Technology. 2012: 418-423

[10] Katerina G P, Trivedi K S. Failure Correlation in Software Reliability Models[J]. IEEE Trans. on Reliability, 2000, 49(1): 37-48

[11] Chen S, Mills S. A binary Markov Process Model for Random Testing[J]. IEEE Trans. on Software Engineering, 1996, 22(3): 218-223

[12] Bishop P G, Pullen F D. PODS Revisited-A Study of Software Failure Behavior[C]//Proc. of the IEEE Int'1 Symp. On Fault Tolerant Computing. Tokio, 1988