软件缺陷分类和分析研究

王 斌 吴太文 胡培培
(中南大学信息科学与工程学院 长沙 410083)

摘 要 缺陷是软件产品的固有成分,如何管理、减少和预防缺陷,对于提高软件质量、降低软件成本具有重要的意义。从缺陷分类和缺陷分析两个方面介绍了软件缺陷研究的现状,对比、分析了各种缺陷分类方法的优势和不足,总结了缺陷分析的主要研究方向及其研究方法,最后对缺陷研究方法的选择进行了讨论。

关键词 软件缺陷,缺陷研究,缺陷分类,缺陷分析,缺陷预测
中图法分类号 TP311.5 **文献标识码** A

Research on Software Defect Classification and Analysis

WANG Bin WU Tai-wen HU Pei-pei
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract Software defect is an inherent component of software products, and how to manage, decrease or prevent defects plays an important role in improving software quality and reducing the cost of projects. This paper introduced software defect from two aspects: defect classification and defect analysis, analyzed the advantages and shortcomings of various defect classifications, and then summarized the main research directions and methods of defect analysis. Finally, the strategy how to select suitable defect research methods was discussed.

Keywords Software defect, Defect research, Defect classification, Defect analysis, Defect prediction

在软件开发过程中,缺陷的产生是不可避免的。缺陷致使软件运行于某一特定条件时出现软件故障,损害了软件的质量,降低了软件的可靠性,是导致软件项目延期、开发成本超支的重要因素。轻微的缺陷仅仅使用户感到不方便或不满意(如响应时间太慢、接口使用困难),而某些严重的缺陷却可能成为软件甚至重大事故的致命隐患。为了保证软件的正常运行,降低项目风险并减少评估软件质量的需要,必须对软件缺陷进行有效的管理和分析。

目前有关软件缺陷研究的综述性资料较为缺乏,不便于研究者了解和认识软件缺陷及其研究现状。基于此,本文系统介绍了软件缺陷的分类方法和应用范畴,以及目前缺陷分析研究的主要方向及其研究方法,从而使研究者对缺陷研究有一个较为全面的认识。

1 软件缺陷概述

1.1 软件缺陷的定义

对于软件缺陷这个概念,目前仍没有一个明确的定义,以致常常与故障、bug、错误、失效等概念混淆。

本文采用IEEE 729-1983(Standard Glossary of Software Engineering Terminology,软件工程术语的标准词汇)[1]对软件缺陷的定义:从产品内部看,缺陷是软件产品开发或维护过程中存在的错误、毛病等各种问题;从产品外部看,缺陷是系统对所需要实现的某种功能的失效或违背。

以下是几个与缺陷密切相关的概念。

故障:在软件运行过程中出现的一种不期望或者不可接受的内部状态。

失效:在软件外部,软件对用户所要求功能的偏离或不满足。

问题:用户在使用软件时,遭遇的困惑或困难。

图1描述了缺陷、故障、失效及问题之间的关联关系。可见,故障、失效及问题都是软件缺陷不同层次的表征,而缺陷是故障、失效和问题产生的根源。每个故障(失效、问题)都是一个缺陷,但并不是每个缺陷都是一个故障(失效、问题)。

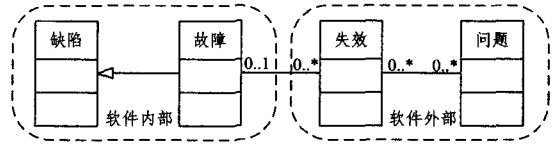

图1 缺陷、故障、失效及问题之间的UML关系图

1.2 缺陷分类和缺陷分析的概念

软件缺陷的研究主要分为两个方面:缺陷分类和缺陷分析。本文对缺陷分类和缺陷分析定义如下。

缺陷分类:对缺陷数据进行属性定义,以便进行分类和管理。

如表1所列,缺陷特征通过8个缺陷属性来描述,缺陷属性则从各个方面反映了缺陷的具体情况。

到稿日期:2012-11-23 返修日期:2013-01-28 本文受国家自然科学基金项目(60970039,61240039)资助。

王 斌(1973-),男,副教授,博士生导师,主要研究方向为软件工程,E-mail:wb_csut@csu.edu.cn;吴太文(1988-),男,硕士生,主要研究方向为软件工程;胡培培(1989-),女,硕士生,主要研究方向为软件工程。

表 1　缺陷属性的定义实例

属性名称	描述
缺陷标识	标记每一个缺陷的符号，具有唯一性
缺陷类型	缺陷种类
缺陷严重程度	因缺陷引起的故障对软件产品的影响程度
缺陷优先级	缺陷必须被修复的紧急程度
缺陷状态	跟踪缺陷修复的进展情况
缺陷起源	引起故障或第一次被检测到所处的软件阶段
缺陷来源	缺陷起因
缺陷根源	引起缺陷的根本原因

缺陷分析：使用一定的方法，定性或定量地挖掘缺陷数据中的隐含价值。

事实上，缺陷分析是一个含义广泛的概念，应用范畴宽广，具有数量繁多的分析方法和分析指标，不同的方法和指标面向不同的度量目标[2]。图 2 给出了缺陷分析的主要应用领域。

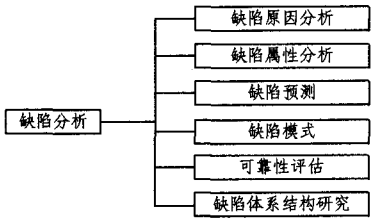

图 2　缺陷分析的主要应用领域

2　缺陷分类的研究

软件缺陷分类的方法繁多，不同的方法因目的不同，其分类过程、复杂程度、准确性（分类是否存在歧义、缺陷类型是否具体）及应用领域也各不相同[3]。缺陷分类使缺陷信息更为丰富、清晰，提高了缺陷数据的收集和管理效率，为缺陷研究奠定了数据基础。

目前业界常用的缺陷管理工具有 Atlassian 公司的 JIRA、Mercury 公司的 Quality Center、IBM 公司的 Rational ClearQuest 等。但这些工具鲜少将缺陷分类方法运用其中，且在缺陷数据的分析方面普遍比较薄弱，通常只提供了一些缺陷属性数量的简单统计功能。

下面是几个有代表性的缺陷分类方法。

2.1　正交缺陷分类法

正交缺陷分类法（Orthogonal Defect Classification，ODC）由 IBM 公司提出，它结合了根本原因分析和统计建模两种软件缺陷分析技术的优势，提供了一套用于捕获缺陷数据关键特性的方案，并就如何对分类的缺陷数据进行分析给予了指导[4]。

ODC 用多个属性来描述缺陷特征，它的最新版本定义了 8 个缺陷属性[5]，对应了缺陷的发现和修复两类特定的活动，如表 2 所列。其中 Defect Type 和 Triggers 属性是该分类方法的基础[3]，Defect Type 反映了缺陷的内在特征，Triggers 则反映了缺陷的形成过程。

ODC 适用于缺陷的定位、排除、原因分析及预防活动，同样，它也可用于改进的软件开发过程。ODC 的缺点在于分类过程复杂，难以把握分类标准，缺陷分析人员的主观意见会影响属性的确定。

目前，ODC 已在许多软件公司和组织中得以运用，包括 IBM、Motorola、Nortel、Lucent 等。业界在使用 ODC 的同时，也在实践中不断对 ODC 进行改进。文献[6]使用贝叶斯网络提高了 ODC 的分类效率；文献[7]提出了一种适用于黑盒测试缺陷的 ODC；文献[8]提出了一种自动化的分类方法 Auto ODC。

表 2　ODC 的缺陷属性定义

属性	译名	说明	引入时机
Activity	活动	缺陷发现的具体活动（单元测试、功能测试等）	发现缺陷
Triggers	触发器	暴露缺陷时存在的环境或者条件	发现缺陷
Impact	影响	对用户或者认识到的、或者实际的影响	发现缺陷
Target	目标	修复缺陷时，涉及实体的高层特征（设计、代码等）	修复缺陷
Defect Type	缺陷类型	缺陷的类型	修复缺陷
Qualifier	限定词	定义了引起缺陷的原因	修复缺陷
Source	来源	缺陷所在的软件开发阶段（内部代码、重用库代码、移植等）	修复缺陷
Age	阶段	造成缺陷的代码是何时开发的（基础代码、重写代码、新增代码）	修复缺陷

2.2　软件异常分类标准

软件异常分类标准（IEEE standard classification for Anomalies 1044-2009）[9] 为软件异常提供一种统一的分类方法，这些异常可能出现在项目中、产品中或者系统生命周期内。分类数据有多种用途，包括缺陷原因分析、项目管理、软件过程改进等。

该分类标准定义了缺陷分类方法（Defect classification）和失效分类方法（Failure classification），提供了大量可供选择的属性类型及其参考取值集合。使用该分类标准时，可根据预期的目的，确定相应的分类过程，包括分类方法的选择（缺陷分类方法或失效分类方法）、作用时间（分类方法在项目生命周期中何时开始、何时结束）、属性类型的选择、属性值的分配等。

该分类标准灵活度高，可针对实际项目的需求进行适当的裁剪或扩展。主要体现在 2 个方面：

1）没有强制规定分类过程；

2）没有为缺陷（失效）属性指定一个强制的取值集合。

正因为如此，如何制定合适的分类过程就成了一个极富挑战和创造性的工作，而属性的取值则有赖于制定者的能力水平和经验积累。

2.3　Thayer 分类法

Thayer 分类法[10,11] 根据错误的性质进行分类，分类的信息源自软件测试和使用中填写和反馈的问题报告。错误信息分为 16 个类：计算错误；逻辑错误；I/O 错误；数据加工错误；操作系统及支持软件错误；配置错误；接口错误；用户需求改变（用户在使用软件后提出软件无法满足新要求产生的错误）；预置数据库错误；全局变量错误；重复错误；文挡错误：需求一致性错误；性质不明错误；人员操作错误；问题（指软件问题报告中提出的需要答复的问题）。在这 16 个类之下，还有 164 个子类。

该分类方法不局限于软件本身的错误，还包括系统软件错误、人员操作错误等，分类详细周全，适用面广，当然分类也比较复杂。由于没有考虑造成缺陷的过程原因，该分类方法不适用于软件过程改进活动。

2.4 Roger 分类法

Roger 分类法[12]根据缺陷引入原因将其分为 12 种类型：不完整或错误的规格(IES)、误解客户需求(MCC)、刻意违背规格(IDS)、违反编程准则(VPS)、错误的数据表示(EDR)、组件接口的不一致性(ICI)、设计逻辑错误(EDL)、不完全或错误的测试(IET)、不准确或错误的文档说明(IID)、编码错误(PLT)、有歧义或不一致的人机接口(HCI)、其他(OTS)。

该分类方法简单实用，但所提供的缺陷信息十分笼统，对缺陷的分析研究帮助有限。表 3 对上述的缺陷分类方法进行了比较。表中，“分类子叶”指缺陷属性或缺陷类型的数量；“分类布局”指分类的层次结构；“灵活性”表示分类过程的自由程度。

表 3　缺陷分类方法的比较

分类方法	ODC	IEEE 1044-2009	Thayer	Roger
目的	改进软件开发过程	定义一个统一的分类方法	测试 & 排除	定位原因
适用范围	大型软件	任意软件	任意软件	中小软件
能否改进软件开发过程?	能	能	否	否
能否识别非软件的缺陷?	否	能	能	否
分类子叶	8+13	视情况而定	16+164	12
分类布局	水平	视情况而定	分层	水平
灵活性	一般	高	低	低

由表 3 可得到 3 点认识：

1)因目的的不同，不存在通用的缺陷分类方法。

2)不少缺陷分类方法在设计时缺乏对软件开发过程的考虑，对非软件原因的缺陷则考虑得更少。

3)缺陷分类方法的灵活性决定了它的适用性。缺陷划分越是明确、细致，其适用性越差。

在选用缺陷分类方法时，软件公司和组织应根据项目的大小、缺陷分类的目的和拥有的资源情况(人力、时限、技术积累等)择优而选，并适当地添加、裁剪缺陷属性，优化分类过程和缺陷类型，使之更好地满足项目需求。

3　缺陷分析的研究

缺陷分析研究涉及的内容广泛，下面介绍几个主要的缺陷分析研究方向及其研究方法。

3.1　缺陷原因分析

缺陷原因分析，即找出导致软件缺陷产生的基本原因和共性原因，分为定性分析和定量分析两种方法[13]。

1)定性分析方法主要使用根本原因分析法(Root Cause Analysis，RCA)，一般的过程包括选择缺陷数据、分析缺陷数据、识别公共原因并提出改进措施 3 个步骤[14]。常用的 RCA 工具有鱼骨图、FTA 原因树、why-why 分析法等。该方法通常在软件项目的每个开发阶段结束后或定期对缺陷进行逐个分析，并产生详尽的分析结果和解决方案。

RCA 特别适合对一些严重程度较高的缺陷进行重点分析。其缺点是消耗资源大、效率低、实时性差，不太适合用于大型项目。

2)定量分析方法[15]的基本思想是缺陷分类(如 ODC)。通过缺陷属性反映有关软件产品和开发过程的信息，然后运用统计分析等手段推断缺陷被引入的原因。基于缺陷分类的定量分析方法以客观的数据统计为基础，实时性好、分析覆盖率高、资源消耗小，适用于大型项目。但该方法的局限性在于缺陷数据的统计结果难以揭示有关人员和管理因素的深层次问题，往往停留在这些问题的近端原因上。

事实证明，ODC 技术的成本只有 RCA 的 1/10。由于定量的缺陷原因分析方法的优势在于大型项目，因此对于中小型项目中可直接使用 RCA；而对于大型项目，则可以定量的缺陷原因分析方法为主，然后对一些特殊情况和个别严重的缺陷使用 RCA，将定量与定性相结合，以在降低成本的同时达到更好的分析效果。

3.2　缺陷属性分析

缺陷属性的值类型分为文本类型和统计类型，见表 4。一般情况下，缺陷数据分析的对象都是统计类型的缺陷属性。而缺陷属性分析又可分为单属性分析和多属性分析。

表 4　缺陷属性的值类型

值类型		描述	例如
文本类型		描述性的文本，没有固定的取值范围	重现步骤、症状描述
统计类型	分类类型	取值来自一个预先定义好的包含有限个元素的集合	缺陷类型、严重级别
	数值类型	包括整数、浮点数、时长等	修复缺陷的耗时、成本

1)单属性分析，即是针对某一缺陷属性进行统计分析。对于一个分类类型的缺陷属性，单属性分析是统计各属性值所对应的缺陷个数及其百分比；对于一个数值类型的属性，单属性分析可计算其均值、总和、最大/最小值、方差、标准差、中位数等。单属性分析结果可用表格、柱状图、饼状图或曲线图直观显示。例如，图 3 是某软件的缺陷模块分布图。

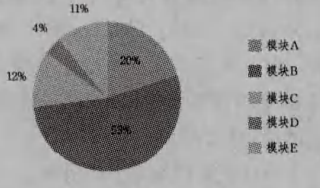

图 3　缺陷模块分布图

2)多属性分析用于揭示多个缺陷属性的取值分布和属性之间的关联关系[16]。多属性的取值分布，也可用表格或图形表示。而多个属性之间的关联分析其方法则因目的而异。文献[16]使用关联规则挖掘算法来发现软件过程中潜在的问题，它同样也可以反映开发活动的效率；文献[17]提出了一种基于主成份分析(Principal Component Analysis，PCA)的多属性分析方法，用于找出项目中软件缺陷的关键原因。

通常，单属性分析得到的信息量有限，多属性分析则能更全面地反映软件过程和技术中存在的问题。在应用中，通过单属性分析发现问题后，仍需进行适当的多属性分析，深度挖掘引发这些变化的根本原因[2]。

3.3　缺陷预测

缺陷预测关注缺陷数量的估计、缺陷的分布、移除和遗留等问题，在分析软件质量、平衡软件成本方面起着重要的作用，大体上分为静态和动态两种缺陷预测方法。静态预测方法，主要是指基于缺陷相关的度量数据对缺陷的数量或者分

布进行预测的方法；而动态方法则是基于缺陷或者失效产生的时间对系统缺陷随时间的分布进行预测的方法。

在软件预测兴起的早期，大多利用统计方法来构建模型[18,19]，此方法存在预测准确性不高、不能获得满意置信区间的问题。鉴于此，研究人员开始利用机器学习的方法建立模型[20,21]，取得了良好的预测效果。基于机器学习的缺陷预测包含3个步骤：

1)选择适当的统计度量元集合，并采集软件产品的统计数据；

2)构造预测模型；

3)将统计数据输入预测模型得到预测结果。

研究表明，使用多个度量元能够更好地表示软件系统的特征，从而更好地预测软件缺陷[22]。而度量元的选取在很大程度上仍然依赖于度量人员的经验。大部分模型在处理不平衡缺陷数据时，预测结果往往与实际不符。文献[23]表明：在缺陷数据不平衡或较为贫乏的情况下，Naïve Bayes模型仍然是一种较好的缺陷预测模型。

当前缺陷预测技术中研究得较多是Bayes模型[24]、CA(Clustering Analysis，聚类分析)[25]、ANN(Artificial Neural Network，神经网络)[26]及SVM(Support Vector Machine，支持向量机)[27]。文献[23,28]对这些技术进行了更全面的描述。

3.4 软件可靠性评估

可靠性是一个关键的软件质量因素，它有两层含义：

1)在规定的条件下、规定的时间内软件不引起系统失效的概率。

2)在规定的时间周期内、在所述的条件下程序执行所要求功能的能力。

软件可靠性评估的基本思路是：基于对软件失效的理解，构建可靠性模型，量化软件可靠性。研究表明，软件失效大多是由软件缺陷引起的。大多数软件可靠性评估模型在数据收集和分析中，都引入了缺陷数据。而缺陷数据必须与过程融合才有价值，因而过程数据也需要收集。通常会将项目的持续时间作为主要关心的过程数据，有时也需细分。

可靠性评估模型大体上可以分为4类[29]：1)失效间隔时间模型；2)缺陷计数模型；3)故障传播模型；4)数据模型。常用的软件可靠性模型有J-M模型、G-O模型、M-O模型、S-Sharp模型等。每个模型都有一定的假设前提、应用范围和限制，且大多数模型也未能得到实际数据的验证[28]。因此，目前还没有一个能适应所有情况的评估模型，也没有任何模型是完备甚至经典的。为此，文献[30]提出了一种以经验数据为基础的最优可靠性模型选择策略。

软件可靠性模型也可以作为软件缺陷预测模型，两者的区别在于：

1)关注点不同。缺陷预测关注于预测结果；而可靠性评估关注于预测结果与实际缺陷情况之间的偏差。

2)软件可靠性评估除了预测模型外，还拥有很多可靠性特征指标，如缺陷密度DD、缺陷纠正率DCE、平均无故障时间MTTF、平均修复时间MTTR等。

3.5 缺陷模式

模式是一种抽象的具名封装，它为特定上下文中的常见问题提供了解决方案，并描述了这种解决方案的结果。而一个软件缺陷模式是指对特定类型的重复或者类似的软件缺陷的抽象描述[31]。

缺陷模式有助于提高软件开发和整体测试的水平。对于开发人员，通过认识缺陷模式，能够有效地预防类似的缺陷；对于测试人员，通过识别缺陷模式，可以更容易交流和修复缺陷。缺陷模式是从实践经验中精炼、抽象而出的缺陷类型，它是一种准确而概括的软件缺陷分类方法。需要说明的是，缺陷分类并不一定源自缺陷模式。

软件缺陷模式的研究主要集中在3个方面：缺陷模式的定义、缺陷模式的整理和基于缺陷模式的应用。目前，已有研究者意识到缺陷模式的重要性，开始关注缺陷模式，并且取得了一定的进展。文献[32]介绍的“Bug patterns in Java”，针对Java语言提出了Bug patterns的定义及相应的缺陷模式。文献[31]将缺陷模式分为3个大类：软件需求缺陷模式、软件设计缺陷模式、软件编码缺陷模式，每个模式下，又进一步细化为许多子模式。文献[33]基于缺陷模式，提出了一种提高软件测试精确度的方法。

3.6 缺陷体系结构研究

在软件演化的过程中，软件结构逐渐偏离原有设计的现象被称为软件体系结构演化。这种结构上的偏离若导致软件质量的下降，则称为软件体系结构退化[34]。软件体系结构退化是软件演化的必然趋势，正如Brooks[35]所说：“系统软件开发是减少混乱度(减少熵)的过程，所以它本身是处于亚稳态的。软件维护是提高混乱度(增加熵)的过程，即使是最熟练的软件维护工作，也只是放缓了系统退化到非稳态的进程”。

软件体系结构退化的研究，可分为诊断、处理、研究、预防4个方面[36]。缺陷体系结构研究是软件体系结构退化研究的一个崭新方向，它从软件缺陷的角度反映了软件体系结构的变化。文献[37]中提出了“缺陷内聚度”和“缺陷耦合度”的概念，以此来度量系统组件的退化程度。将涉及较多数量多组件缺陷(Multiple-Component Defect，MCD)的组件称为与组件退化相关的关键组件，并给出了一种基于缺陷分析识别关键组件的方法。文献[38,39]进一步提出，多组件缺陷与横切关注点有关，由于MCD的修复涉及到对多个组件的变更，修复MCD难度的增加通常预示着软件体系结构复杂度的增加。因此，多组件缺陷也被称为体系结构缺陷。文献[40]系统分析概括了现有技术，提出了一种基于软件缺陷的体系结构退化诊断方法DAD(Diagnosing Architectural Degeneration)。

软件系统的多样性，导致了缺陷状况的差异性。对于不同的软件系统而言，其缺陷状况都是独一无二的。因此，各种缺陷分析的应用技术，能适用于特定的环境，但难以具备普遍的适用性。

结束语 除了本文介绍的缺陷分类方法和缺陷分析应用技术外，缺陷研究还有许多其它的内容，如软件库挖掘[41]等。软件工程活动可分为3个阶段，分别是开发过程、管理过程和过程改进。缺陷研究参与了软件的开发过程和过程改进，但对管理过程的反映力度往往不足，难以体现人在软件工程中的影响作用。高品质的软件需要优良的设计、科学的管理，同

样也不可或缺对软件缺陷的重视和利用。软件缺陷研究虽然消耗了不少资源，但对于软件开发过程的监督和改良、减少和预防软件缺陷、项目经验的积累，贡献是显而易见的。

缺陷研究（缺陷分类、缺陷分析）方法之间并不是相互孤立的，而是彼此关联的，例如：缺陷分析应用技术对缺陷分类方法的强依赖性；缺陷模式对缺陷分类方法的帮助作用；缺陷预测与软件可靠性评估的相互促进关系等。不仅如此，缺陷研究的各类方法有一个共同点：均受到不同的条件或者应用限制。软件系统因需求而产生，并不存在一个万能的框架系统，因而每个软件系统的缺陷状况各不相同，也就不可能存在一种通用的缺陷研究方法。所以，缺陷研究只能因地制宜，借鉴已有的方法，而不能完全照搬。

参 考 文 献

[1] IEEE Std 729-1983[S]. Standard Glossary of Software Engineering Terminology. IEEE, 1990

[2] 闫振兴，郑骏. 软件缺陷度量与分析技术研究[J]. 计算机应用与软件，2011，28(9)：130-133

[3] Ploski J, Rohr M, Schwenkenberg P. Research issues in software fault categorization[J]. ACM SIGSOFT Software Engineering Notes, 2007, 32(6): 1-8

[4] Chillarege R, Bhandari I S, Chaar J K, et al. Orthogonal defect classification a concept for in-process measurements[J]. IEEE Transactions on Software Engineering, 1992, 18(11): 943-956

[5] IBM Research Center for Software Engineering. Orthogonal Defect Classification[OL]. http://www.research.ibm.com/soft-eng/ODC/ODC.HTM, 2002

[6] Wang H, Wang H, Lin Z Q. Improving Classification Efficiency of Orthogonal Defect Classification Via a Bayesian Network Approach[C]// Computational Intelligence and Software Engineering. 2009: 1-4

[7] Li Ning, Li Zhan-huai, Sun Xi-ling. Classification of Software Defect Detected by Black-box Testing: An Empirical Study[C]// Second WRI World Congress on Software Engineering. 2010: 234-240

[8] Huang Li-guo, Ng V, Persing I. AutoODC: Automated Generation of Orthogonal Defect Classifications[C]// Automated Software Engineering(ASE). 2011: 412-415

[9] IEEE Std 1044-2009[S]. Standard Classification for Anomalies. IEEE, 2009

[10] 聂林，刘孟仁. 软件缺陷分类的研究[J]. 计算机应用研究，2004，1(6)：84-86

[11] 黄锡滋. 软件可靠性、安全性与质量保证[M]. 北京：电子工业出版社，2002

[12] Pressman R S. Software Engineering: a Practitioner's Approach (5th)[M]. Thomas Casson, 2001: 209-212

[13] 刘海，郝克刚. 软件缺陷原因分析方法[J]. 计算机科学，2009(1)：242-243，251

[14] Leszak M, Perry D E, Stoll D. A Case Study in Root Cause Analysis[C]// Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland, 2000

[15] Chillarege R. ODC-a 10x for Root Cause Analysis [C]// Proceedings of RAM 2006 Workshop. Berkeley California, 2006

[16] 刘海，郝克刚. 软件缺陷数据的分析方法及其实现[J]. 计算机科学，2008，35(8)：262-264

[17] Song Yu, Wang Xin-hong. Research on Application of Software Defect Analysis based on PCA[C]// 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). 2010

[18] Khoshgoftaar T, Gao K, Szabo R M. An application of zeroinflatedpoisson regression for software fault prediction[C]// The 12th International Symposium on Software Reliability Engineering. 2001: 66-73

[19] Ohlsson N, Zhao M, Helander M. Application of multivariate analysis for software fault prediction[J]. Software Quality Journal, 1998, 7(1): 51

[20] Gondra I. Applying machine learning to software fault-proneness prediction[J]. Journal of Systems and Software, 2008, 81(2): 186-195

[21] Andre B C, Aurora P, Silvia R V. A symbolic fault-prediction model based on multiobjective particle swarm optimization[J]. Journal of Systems and Software, 2010, 83(5): 868-882

[22] Munson J C, Khoshgoftaar T M. The Detection of Fault-Prone Programs[J]. IEEE Transactions on Software Engineering, 1992, 18(5): 423-433

[23] Catal C. Software fault prediction A literature review and current trends[J]. Expert Systems with Applications, 2011, 38: 4626-4636

[24] Turhan B, Bener A. Analysis of Naive Bayes' assumptions on software fault data: An empirical study[J]. Data Knowledge Engineering, 2009, 68(2): 278-290

[25] Mahaweerawat A, Sophatsathit P, Lursinsap C. Adaptive self-organizing map clustering for software fault prediction[C]// Fourth international joint conference on computer science and software engineering. KhonKaen, Thailand, 2007: 35-41

[26] Yang B, Yao L, Huang H Z. Early software quality prediction based on a fuzzy neural network model[C]// Third international conference on natural computation. Haikou, China, 2007: 760-764

[27] 姜慧研，宗茂，刘相莹. 基于 ACO-SVM 的软件缺陷预测模型的研究[J]. 软件学报，2011，34(6)：1148-1153

[28] 王青，伍书剑，李明树. 软件缺陷预测技术[J]. 软件学报，2008，19(7)：1565-1580

[29] Lyu M R. Handbook of Software Reliability Engineering [M]. IEEE Computer Society Press, 1996

[30] Stringfellow C, Andrews A. An empirical method for selecting software reliability growth models[J]. Empirical Software Engineering, 2002, 7(4): 319-343

[31] Zeng F P, Chen A Z, Tao X. Study on Software Reliability Design Criteria Based on Defect Patterns[C]// The Eighth International Conference onReliability, Maintainability and Safety. Beijing, China, 2009: 723-727

[32] Allen, Eric. Bug Patterns in Java [M]. New York Inc: Springer-Verlag, 2005

[33] Jin Da-hai, Gong Yun-zhan, Xiao Qing. A Method of Improving Precision in Software Testing Based on Defect Patterns[C]// The International Conference on Industrial and Information Systems. 2009: 285-288

（下转第 24 页）

$V(CT_n)=\{1,2,\cdots,n\}$，边集 $E(CT_n)=\{(i,j):1\leqslant i<j\leqslant n\}$，$k$ 为正整数，则称 (n,k)-多部 Cayley 图 (n,k)-$\Gamma(CT_n)$ 为 (n,k)-多部完全对换网络，仍记为 (n,k)-$\Gamma(CT_n)$。CT_n 的 Cayley 图即为著名的完全对换网络（complete transposition network）[9]。

由定理 1 得到：

推论 6 1) $(n,1)$-多部轮网络 $(n,1)$-$\Gamma(CT_n)$ 就是完全对换网络，它的结点数为 $n!$，结点度为 $n(n-1)/2$，直径为 $n-1$，也是连通图，也是 Cayley 图，也是点可迁的，点连通度和边连通度均为 $n(n-1)/2$。

2) $k\geqslant 2$ 时，(n,k)-$\Gamma(CT_n)$ 的结点数为 $kn!$，结点度为 $n(n-1)$，直径为 $n+k-2$，也是 Cayley 图，也是点可迁的。

由此可见，当 $k\geqslant 2$ 时，(n,k)-$\Gamma(CT_n)$ 也有很好的可扩展性，当 n 不变时，直径随 k 增长缓慢。

结束语 对超级计算机的发展来讲，互连网络的设计是一个永恒的主题。因为互连网络的性能指标之间存在着矛盾，例如，小的结点度和高的连通度（容错性）之间总是一对矛盾，这为互连网络的设计提供了广阔的空间。在本文中，我们提出了互连网络的 (n,k)-多部 Cayley 图模型，设计出了多种互连网络，并讨论了它们的基本性质，证明它们有很好的可扩展性。它们的其他性能有待进一步的研究。

参 考 文 献

[1] Akers S B, Krishnamurthy B. A group-theoretic model for interconnection networks [J]. IEEE Transactions on computers, 1989, 38(4): 555-565

[2] Bondy J A, Murty U S R. Graph Theory with Applications[M]. London: Macmillian Press, 1976

[3] 徐俊明. 组合网络理论[M]. 北京：科学出版社，2007

[4] 高随祥. 图论与网络流理论[M]. 北京：高等教育出版社，2009

[5] 王鼎兴，陈国良. 互连网络结构分析[M]. 北京：科学出版社，1990

[6] Huang Kai. 高等计算机系统结构[M]. 王鼎兴，等译. 北京：清华大学出版社，南宁：广西科学技术出版社，1992

[7] Du D Z, Hsu D F, et al. Combinatorial Network Theory[M]. Kluwer Academic Publishers, 1996: 65-105

[8] 师海忠. 互连网络的代数环模型[D]. 北京：中国科学院应用数学研究所，1998

[9] Lakshmivarahan S, Jwo J S, Dhall S. Symmetric in interconnection networks based on Cayley graph of permutation groups: A survey[J]. Parallel Computing, 1993, 19(4): 361-407

[10] 师海忠，牛攀峰，马继勇，等. 互连网络的向量图模型[J]. 运筹学学报，2011，15(3)：115-123

[11] 师海忠，路建波. 关于互连网络的几个猜想[J]. 计算机工程与应用，2008，44(31)：112-115

[12] 师海忠，马继勇，牛攀峰，等. 关于修正冒泡排序网络的一簇猜想[J]. 计算机科学，2011，2011，38(10A)：265-267

[13] 师海忠，王国亮，马继勇，等. 完全对换网络的一簇猜想[J]. 计算机科学，2012，39(6A)：404-407

[14] 路建波，师海忠，牛攀峰. Star 网络 S_6 的 Hamilton 圈分解[J]. 工程数学学报，2011，28(1)：565-568

[15] Walker D, Latifi S. Improving bounds on link failure tolerance of the star graph[J]. Information Sciences, 2010, 180(13): 2571-2575

[16] Wan Min, Zhang Zhao. Akind of condition vertex connectivity of star graphs[J]. Applied Mathematics Letters, 2009, 22(2): 264-267

[17] Wang Jian, Xu Xi-rong, Zhu De-jun, et al. On the bounds of feedback numbers of (n,k)-star graphs[J]. Information Processing Letters, 2012, 112(12): 473-478

[18] Tsai Ping-ying, Fu Jung-sheng, Chen Gen-huey. Fault-free longest paths in star networks with conditional link faults[J]. Theoretical Computer Science, 2009, 410(8-10): 766-775

[19] Li T-K, Tan J J M, Hsu L-H. Hyper Hamiltonian laceability on edge fault star graph[J]. Information Sciences, 2004, 165(1/2): 59-71

[20] Rouskov Y, Latifi S, Srimani P K. Conditional fault diameter of star graph networks[J]. Journal of Parallel and Distributed computing, 1996, 33(1): 91-97

[21] Biggs N. Algebraic Graph Theory[M]. MA: Cambridge University Press, 1993

[22] 张禾瑞. 近世代数基础[M]. 北京：高等教育出版社，1978

[23] Chou Z-T, Hsu C-C, Shen Jang-ping. Bubblesort star graphs: A New Interconnection Networks[C]// Proceeding of the 1996 International Conference on Parallel and Distributed Systems. 1996

[24] Cheng E, Liptak L. Fault resiliency of Cayley graphs generated by transposition [J]. International Journal of Foundations of Computer Science, 2007, 18(5): 1005-1022

（上接第 20 页）

[34] Perry D E, Wolf A L. Foundations for the study of software architecture[J]. ACM SIGSOFT Software Engineering Notes, 1992, 17(4): 40-52

[35] Brooks F P Jr. The Mythical Man-Month[M]. Posts & Telecom Press, 2010

[36] Hochstein L, Lindvall M. Combating architectural degeneration: a survey[J]. Information and Software Technology, 2005, 47(10): 643-656

[37] Andrews A A, Ohlsson M C, Wohlin C. Deriving fault architectures from defect history[J]. Journal of Software Maintenance: Research and Practice, 2000, 12(5): 287-304

[38] Li Zu-de, Gittens M, Murtaza S S, et al. Analysis of pervasive multiple-component defects in a large software system[C]// Proceedings of 2009 IEEE International Conference on Software Maintenance(ICSM'09). 2009: 265-273

[39] Eaddy M, Zimmermann T, Sherwood K D, et al. Do crosscutting concerns cause defects[J]. IEEE Transactions on Software Engineering, 2008, 34(4): 497-515

[40] Li Zu-de. Characterizing and Diagnosing Architectural Degeneration of Software System from Defect Perspective[OL]. http://ir.lib.uwo.ca/etd/30/, 2010

[41] 刘英博，王建民. 面向缺陷分析的软件库方法综述[J]. 计算机科学，2007，34(9)：1-11