第32卷 第19期 计 算 机 工 程 2006年10月
Vol.32 № 19 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)19—0088—03 文献标识码：A 中图分类号：TP311

软件缺陷的综合研究

梁成才[1]，章代雨[2]，林海静[1]
(1. 华东计算技术研究所，上海 200233；2. 空军驻上海地区军事代表室，上海 200233)

摘 要：软件缺陷的概念在软件质量范畴中处于举足轻重的地位，软件缺陷度量是软件质量度量范畴内的核心度量。该文区分了错误、缺陷、故障、失效4个软件缺陷相关的概念，采用正交缺陷分类法建立了软件缺陷的分类分级模式，剖析了软件缺陷的生存周期，给出了缺陷密度、缺陷泄漏矩阵、缺陷注入率和缺陷消除率等基本的、实用的软件缺陷度量。

关键词：软件缺陷；缺陷分类分级；缺陷注入；缺陷清除；缺陷密度

Integrated Research of Software Defect

LIANG Chengcai[1], ZHANG Daiyu[2], LIN Haijing[1]
(1. East China Institute of Computer Technology, Shanghai 200233;
2. Air Force Military Representative Office in Shanghai Area, Shanghai 200233)

【Abstract】 Software defect is the most important concept in software quality category, and software defect measurement is the core measurement in software quality measurement category. The related concepts of software defect such as error, defect, fault and failure are discriminated. The classifying model and severity-level grading model of software defect are established by using the orthogonal defect classification method, the life cycle of software defect is analyzed. The basic and practical metrics of software defect such as defect density, defect leakage matrix, defect injection rate and defect removal rate are given.

【Key words】 Software defect; Defect classification & severity-level gradation; Defect injection; Defect removal; Defect density

质量、进度和成本是软件项目关注的3大要素，它们相互依赖、相互制约，软件项目管理就是要在这3个目标的建立、跟踪和实现方面达成最佳均衡。软件质量要求的高低，既会影响到成本，又会影响到进度。一般来说，不论是顾客还是承制方，对软件质量的既定目标都不会妥协，宁可追加成本或延迟进度，也不会迁就质量目标的降低，对质量都采用“一票否决制”，因此，软件质量在三大目标中往往是位于第一位的。不论是业界传统的狭义软件质量观还是以ISO/IEC 9126系列标准代表的广义软件质量观，软件缺陷的概念都在软件质量范畴中处于举足轻重的地位。在狭义软件质量观中，软件缺陷的概念绝对处于核心地位，它衍生出一系列软件质量指标，是考察软件质量的唯一依据。在当今业界大力实施的软件能力成熟度模型CMM(Capability Maturity Model for Software)或能力成熟度模型集成CMMI(Capability Maturity Model Integration)中，软件缺陷度量仍然是一个核心度量，尤其是攀登高成熟度级别不可或缺的基石。总之，软件缺陷度量的体系结构是各种软件过程改进模型的基础设施，是实施与评估软件质量活动的先决条件。建立完善的软件缺陷度量的体系结构对软件质量管理的策划和实施、软件缺陷的跟踪和管理、软件质量目标的设定和实现、软件过程能力基线的创建和优化都具有重大的现实意义。

1 相关概念的区分

历史上，曾经对与软件缺陷相关的一些概念搞得很混淆，各家众说纷纭，使得从业人员无所适从。下面对这些概念进行区分，它们是建立软件缺陷度量的体系结构的先决条件。

(1)错误(error)：是人为错误，指软件开发人员在开发软件的过程中无意间犯下的技术错误，正是这些错误导致软件工作产品的缺陷。

(2)缺陷(defect)：是软件工作产品中不满足指定要求的成分，它是静态的，如果不将其消除，它将永远存在。在业界，人们常用另外一个词“Bug”指代缺陷，这是从早期美国海军在调试软件时一个臭虫(bug)引发了系统不能正常工作的典故所流传下来的将“Bug”作为缺陷的代名词的原因。将缺陷俗称为“Bug”，易使人对缺陷轻描淡写，忽视了缺陷的严重性，这是值得注意的问题。缺陷是造成软件故障乃至失效的内在原因。

(3)故障(fault)：是软件运行时丧失了在规定的限度内执行所需功能的能力，它是动态的，它可能导致失效。故障不一定导致软件失效，软件运行可以出现故障但不出现失效，如在容错(fault tolerance)软件运行中容许有规定数量的故障出现而不导致失效。对无容错的软件，故障即失效。故障是软件缺陷的外在表现。

(4)失效(failure)：是软件运行时不能完成规定功能，它是动态的，由故障所导致。失效是软件缺陷的外在表现。

以上描述的软件失效机制如图1所示。

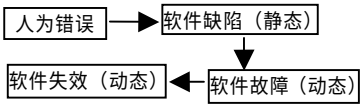

图1 软件失效机制示意图

作者简介：梁成才(1965－)，男，高工，主研方向：软件质量管理和软件测试；章代雨，高工；林海静，助工

收稿日期：2006-08-07 **E-mail：**lcc@ecict.com

2 软件缺陷的分类分级

历史上，出于对软件缺陷管理的需要，人们曾经对软件缺陷进行过分类分级，如 IBM 公司等软件开发大公司都制定了内部的软件缺陷分类分级的企业标准，企业标准主要依据本企业的实际情况制定，目的是为本企业所用。另外，还出现过一些推荐性的工业标准，如国军标 GJB 2255－94。但是从未出现过权威性、实用性、系统性的软件缺陷分类分级的标准，就是大多数企业标准或推荐性的工业标准也都主要关注软件代码的缺陷分类分级，没有覆盖其它主要软件工作产品的缺陷情况。本节试图对软件缺陷进行系统性、实用性的分类分级，包括划分软件缺陷的类型、类别和严重性等级，它是建立软件缺陷度量的体系结构的基础。

采用正交缺陷分类(orthogonal defect classification)法划分覆盖软件需求、软件设计和软件代码等主要软件工作产品的软件缺陷类型如表 1 所示。

表 1 软件缺陷类型

大类号	大类名称	小类号	小类名称
1	需求缺陷 (软件需求不满足顾客的要求)	1	功能
		2	性能
		3	接口
		4	控制流
		5	数据流
		6	标准
		7	一致性
		8	可追溯性
		9	文档版本
		10	其它
2	设计缺陷 (软件设计不满足软件需求规格说明的要求)	1	功能
		2	性能
		3	接口
		4	逻辑
		5	数据使用
		6	错误处理
		7	标准
		8	一致性
		9	可追溯性
		10	文档版本
		11	其它
3	代码缺陷 (软件代码不满足软件设计说明的要求)	1	功能
		2	性能
		3	接口
		4	逻辑
		5	数据使用
		6	错误处理
		7	编程语言
		8	编程规范
		9	软件多余物
		10	可追溯性
		11	代码版本
		12	其它
4	使用文档缺陷	1	完备性
		2	一致性
		3	正确性
		4	文档版本
		5	其它

软件缺陷类别用于区分软件缺陷的性质，对软件缺陷类别的划分如表 2 所示。

软件缺陷造成软件故障和软件失效的严重程度也不一样，对用户带来的危害和损失程度不一样，因此有必要区分软件缺陷的严重等级，对软件缺陷严重等级的划分见表 3。

表 2 软件缺陷类别

序号	类别名称	说明
1	遗漏	应有的内容缺失
2	错误	相应的内容错误
3	多余	不应有的内容出现

表 3 软件缺陷严重等级

序号	等级名称	说明
1	致命	导致软件崩溃、死机的缺陷
2	严重	妨碍主要功能实现和性能达标的缺陷
3	一般	妨碍次要功能实现的缺陷
4	轻微	给用户带来不方便或麻烦、但是不妨碍功能实现的缺陷

3 软件缺陷的生存周期

软件缺陷的生存周期是指软件缺陷从产生到消除的持续时间。当由于人工失误将缺陷注入(inject)软件工作产品中后，在软件产品发布前发现软件缺陷的主要方法是软件评审和软件测试。在软件生存周期中，各个阶段注入的软件缺陷被发现和消除(remove)的时机包括本阶段以及此后的所有阶段，每个阶段注入的软件缺陷在本阶段被发现和消除是最佳的选择，因为前期阶段注入的软件缺陷在后面阶段被发现和消除的代价比在本阶段被发现和消除所花的代价高得多，势必造成成本目标实现的不利风险。

在软件生存周期的每个阶段，软件质量控制的目标是既要尽量发现和消除本阶段的软件缺陷，还要尽量发现和消除以前所有阶段遗留下来的软件缺陷。在软件生存周期的某个阶段软件缺陷的注入、发现和消除机制如图 2 所示。

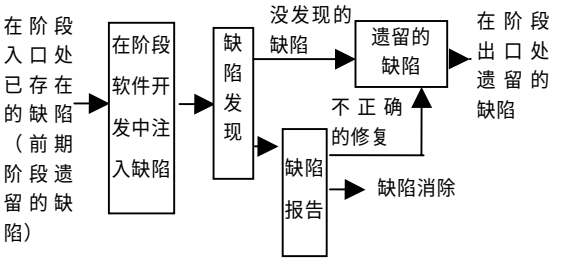

图 2 软件生存周期阶段的软件缺陷注入、发现和消除机制

4 软件缺陷的基本度量

4.1 缺陷密度

基于软件缺陷的软件质量的一个事实上的标准度量是缺陷密度。对于一个软件工作产品，软件缺陷分为两种：通过评审或测试等方法发现的已知缺陷(known defect)、尚未发现的潜在缺陷(latent defect)。缺陷密度的定义如下：

$$\text{缺陷密度} = \frac{\text{已知缺陷的数量}}{\text{产品规模}}$$

在缺陷密度的公式中，产品规模的度量单位可以是文档页、代码行、功能点。

缺陷密度是软件缺陷的基本度量，可用于设定产品质量目标、支持软件可靠性模型(如 Rayleigh 模型)预测潜伏的软件缺陷进而对软件质量进行跟踪和管理、支持基于缺陷计数的软件可靠性增长模型(如 Musa-Okumoto 模型)对软件质量

目标进行跟踪并评判能否结束软件测试。

4.2 缺陷注入率和缺陷消除率

结合软件缺陷的生存周期，可以提出另外一对十分有用的软件缺陷度量：缺陷注入率和缺陷消除率。为此，须先从软件缺陷的生存周期引出一个重要概念：缺陷泄漏矩阵(defect leakage matrix)也称缺陷起源-发现处矩阵(defect origin-where found matrix)。

4.2.1 缺陷泄漏矩阵

缺陷泄漏矩阵是一个下三角矩阵，矩阵的行是软件缺陷的发现处(发现阶段)，矩阵的列是软件缺陷的起源阶段，其元素N_{ij}满足下列条件：

(1)只有当i≥j的元素N_{ij}才有数据；

(2)对角线(元素N_{ij}，当i=j)上包含阶段注入并在同一个阶段被发现和清除的缺陷数量；

(3)对角线以下(元素N_{ij}，当i>j)的元素包含来自早期阶段而后来才被发现和清除的缺陷数量；

(4)对角线以上(元素N_{ij}，当i<j)的元素是空的，因为早期阶段不可能发现后来阶段注入的缺陷。

缺陷泄漏矩阵如图 3 所示。

缺陷起源

发现处	j=1	j=2	j=3	*	*	j	*	j=K	行合计
i=1	N_{11}								F_1
i=2	N_{21}	N_{22}							F_2
i=3	N_{31}	N_{32}	N_{33}						F_3
*	*	*	*	*					*
*	*	*	*	N_{ij} (i>j)	*				*
i	*	*	*	*	*	N_{ij} (i=j)			F_i
*	*	*	*	*	*	*	*		*
i=K	N_{K1}	N_{K2}	N_{K3}	*	*	*	*	N_{KK}	F_K
列合计	O_1	O_2	O_3	*	*	O_j	*	O_K	T

图 3 缺陷泄漏矩阵示例

在图 3 中，在缺陷泄漏矩阵的右方和下方各加了“行合计”和“列合计”两条边，两条边的元素分别计算如下：

$$F_i = \sum_{j=1}^{i} N_{ij}, i = 1, \ldots, K$$

Fi 表示第 i 阶段发现和清除的缺陷总数量。

$$O_j = \sum_{i=j}^{K} N_{ij}, j = 1, \ldots, K$$

Oj 表示第 j 阶段注入的已知缺陷总数量。

“行合计”和“列合计”两条边的交汇处是符号 T(Total 的首字母)，表示在迄今为止经历的软件生存周期的各个阶段中所有发现和清除的缺陷总数量。T 的计算公式如下：

$$T = \sum_{i=1}^{K} F_i \text{ 或 } T = \sum_{j=1}^{K} O_j$$

4.2.2 缺陷注入率

缺陷注入率 DIR(Defect Injection Rate)表示在迄今为止经历的软件生存周期的各个阶段中每个阶段注入缺陷的比率。DIR 的计算公式如下：

$$DIR_i = \frac{O_i}{T}, i = 1, \ldots, K$$

4.2.3 缺陷消除率

缺陷消除率 DRR(Defect Removal Rate)表示在迄今为止经历的软件生存周期的各个阶段中每个阶段消除缺陷的比率。

缺陷消除率有两种计算公式，一种公式反映了“各人自扫门前雪”的情况，即各个阶段如何有效地发现和清除本阶段注入的缺陷数量，其计算公式如下：

$$DRR1_i = \frac{N_{ii}}{O_i}, i = 1, \ldots, K$$

缺陷消除率的另一种公式反映了每个阶段对软件质量控制整体的贡献，即各个阶段如何有效地发现和清除迄今为止软件存在的所有缺陷，包括本阶段注入的缺陷和以往阶段遗留下来的缺陷，其计算公式如下：

$$DRR2_i = \frac{F_i}{\sum_{j=1}^{i} O_j - \sum_{j=1}^{i-1} F_j}, i = 2, \ldots, K$$

使用缺陷注入率和缺陷消除率可以对软件开发过程进行有效性考察，并创建软件过程能力基线进行软件过程改进。因此，它们是实行 CMM 或 CMMI 定量管理级的基本工具。

5 小结

本文探讨了与软件缺陷度量的体系结构密切相关的一些问题，包括软件缺陷相关概念之间的区别、软件缺陷的分类分级模式、软件缺陷的生存周期以及基本的软件缺陷度量，它们都是在软件质量管理理论和实践中易混淆、常误用、欠规范、难操作的棘手问题，是实施有效的、量化的软件质量管理的几大障碍，本文试图对上述问题进行剖析和澄清，供业界同仁在软件过程改进和软件质量管理中参考。

参考文献

1 吴明晖, 应　晶译. 软件质量工程[M]. 北京: 电子工业出版社, 2004-07.

2 杨海燕, 赵　巍, 张　力等译. 软件度量[M]. 北京: 机械工业出版社, 2004-09.

3 何国伟, 王　纬. 软件可靠性[M]. 北京: 国防工业出版社, 1998-01.

4 中国人民解放军总装备部. GJB 5236-2004 军用软件质量度量[S]. 2004.

5 李怀璋, 武占春, 王　青等译. 软件质量保证[M]. 北京: 机械工业出版社, 2003-05.

6 吴超英, 廖彬山译. 实用软件度量[M]. 北京: 机械工业出版社, 2003-09.

7 郑人杰, 王　纬, 王方德等. 基于软件能力成熟度模型(CMM)的软件过程改进[M]. 北京: 清华大学出版社, 2003-03.

8 中国人民解放军总装备部. GJB 5000-2003 军用软件能力成熟度模型[S]. 2003.

9 刘孟仁译. 能力成熟度模型(CMM): 软件过程改进指南[M]. 北京: 电子工业出版社, 2001-07.

10 中国人民解放军总装备部. GJB 2434A-2004 军用软件产品评价[S]. 2004.

11 韩　柯译. 软件可靠性工程[M]. 北京: 机械工业出版社, 2003.

12 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, 2002-10.

13 方德英译. IT 度量[M]. 北京: 清华大学出版社, 2003-12.

14 中华人民共和国国家质量监督校验检疫总局. GB/T 8566-2001 信息技术软件生存周期过程[S]. 2001.