软件缺陷移除效率及其测量

Software Defect Removal Effectiveness and Its Measurement

韩婷婷 黎翠凤 （鲁东大学计算机科学与技术学院 山东烟台 264025）

摘要：软件缺陷移除在任何软件项目中都是开销最大的活动，并在很大程度上影响项目进度，因而对缺陷移除效率的测量成为软件开发组织最为重要的一项软件质量度量。为此通过矩阵的方法来记录缺陷起源和缺陷发现阶段的数据，得出了缺陷移除效率的操作定义，并定义了阶段缺陷移除效率、全面缺陷移除效率、审查效率和测试效率的计算公式。

关键词：缺陷移除效率 操作定义 千行代码 阶段缺陷移除效率 全面缺陷移除效率 全面审查效率 全面测试效率

1 引言

20 世纪 90 年代以来软件开发进入到了一个质量的时代，社会发展对软件的日益依赖进一步强化了对于软件质量的要求。在这样一个质量时代，不断提高软件产品质量和软件过程质量已经成为每个软件开发组织的首要目标。比较类似的软件项目可以发现，成功的项目往往在软件质量控制方面取得了很大成绩，这取决于项目过程中良好的软件质量度量。软件产品质量的一个重要度量是软件的缺陷密度（缺陷率），软件开发组织通过缺陷移除来降低缺陷密度并提高软件质量。测量软件缺陷移除效率可以衡量软件开发组织有效移除缺陷提高软件质量的能力，并且有助于确定开发过程中哪些部分有最大的改进机会，从而对软件开发过程采取措施进行改进。综上，对于所有项目和开发组织而言，缺陷移除效率（defect removal effectiveness）的概念及其测量成为了软件开发的中心问题。

2 测量理论基础

测量是科学领域探索活动的基础，近代的自然科学都是从真正意义上的测量开始的。测量在软件这个充满问题的复杂学科中起到了关键的作用，同时也为软件工程成为一门真正的工程学科提供了科学的基础。同其它学科一样，软件理论的基础也是概念和定义，为了证实或否定一个说法，首先需要定义该说法中的关键概念，并通过命题描述概念之间的关系。根据每个命题，可以得出一个或多个经验假设，为了能够得到可以通过测量和数据证实的假设，对于概念的一般化定义是不够的，需要给出概念具体化定义的含义，这种具体化的定义称为操作定义（operational definition），说明了度量和得到数据的过程[1]，例如："身高"的一个操作定义应当说明如何测量被测者的身高，使用何种测量仪器，以及记录结果的测量单位。"软件产品缺陷密度"的操作定义应当说明缺陷密度的公式，说明公式各部分的含义，如何测量，测量单位等。在概念的操作定义的基础上就可以在现实世界中收集数据并测试这些假设。最后，可以对测量结果运用统计学的知识进行分析。假如数据支持该假设，那么就对该假设做出肯定；否则，就否定该假设。对于肯定的假设，可以将其作为一定得措施在软件开发中进行实施，相应地提高软件开发质量。

3 软件产品质量度量

软件质量度量是软件度量的一个子集，它侧重于产品、过程和项目的质量细节[1]。从整个软件生命周期的角度来看，软件质量度量可以进一步分为：产品质量度量、过程质量度量和维护质量度量。其中产品质量度量是从产品本质质量和用户满意度两个方面对软件产品进行直接的度量，是软件质量度量的一个核心分支。

软件质量的定义包括两个层次，第一是产品的内在质量，通常用平均无故障时间和缺陷密度来测量；第二层是广义的质量，包括用户满意度和用户问题。对于产品的内在质量，没有缺陷和拥有良好的可靠性是最基本的要求。平均无故障时间（MTTF）常用于安全性较高的系统，例如：航空控制系统。由于产品的缺陷数目对软件维护阶段费用和资源预测是非常重要的，因而缺陷密度度量对于商业软件开发机构更具有吸引力。

3.1 缺陷密度度量

缺陷率的通用概念是一定时间范围内的缺陷数与错误几率的比值，缺陷率的分母是软件的大小，通常用千行代码（KLOC）数或功能点数来表示[1]。由于实际使用的编程语言的差异，使得 LOC 计数比较复杂。对于同一种语言，实际的代码行数的计数也有不同的方法。大致包括实际 LOC 和逻辑 LOC 两种计数方法。实际 LOC 可以包括程序头、数据定义，甚至可以包括注释。逻辑 LOC 包括逻辑代码行，不包括注释和程序头。实际的编程语言、编码风格和计算方法都会影响 LOC 的计算，所以在计算缺陷率的时候，应该给出 LOC 计算的说明。

4 缺陷移除效率

缺陷移除是整个软件开发和维护期间开销最大的一项活动，有效的缺陷移除可以帮助开发组织在很大程度上改进软件质量、降低开发成本和达到预期进度。在整个软件生命周期中，缺陷的数目在很大程度上是由维护阶段之前的开发过程缺陷移除决定的，在维护过程中，并不能很大程度地提高产品质量。因而，对于开发组织而言，测量项目全面缺陷移除过程的效率以及阶段缺陷移除效率是至关重要的。

4.1 缺陷移除效率的文献回顾

20 世纪 50 至 60 年代的程序设计阶段，唯一的缺陷移除手段就是软件测试。但是测试往往是软件开发的下游环节，这使得软件项目为缺陷移除付出了巨大的代价。20 世纪 70 年代，正式的评审和审查被用于进行缺陷移除，提供了项目早期阶段的缺陷移除手段。在 Fagan（1976）关于设计和代码审查的文章中，涉及了缺陷移除效率的概念[1]。他定义了错误发现效率为

$$\frac{\text{一次审次审查发现的}}{\text{审查前产品中的总错误数}} \times 100\%$$

Robert Dunn 对个别活动的有效性的简单度量公式是：

$$E=\frac{N}{N+S}\times 100\%$$

其中 E = 活动的效率，N = 活动中发现的故障（缺陷）数目，S = 后续活动中发现的故障（缺陷）数目。这个度量可以通过仅选择那些在活动期间出现的和对活动检测敏感的故障（缺陷），以对此测量进行调整[2]。

4.2 缺陷移除效率的操作定义及测量

为了能够对项目的缺陷移除效率进行测量，必须通过进一步分析给出缺陷移除效率的操作定义，从中得到具体的测量方法。

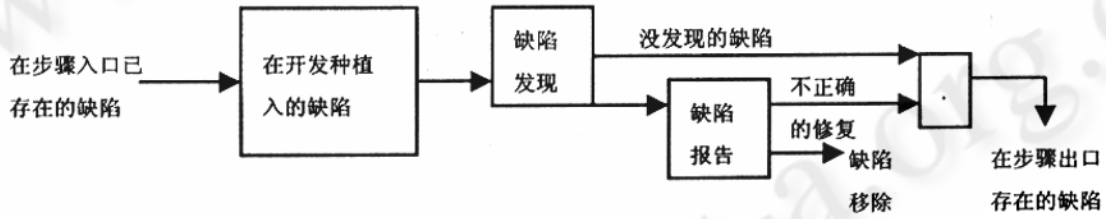

图 1　一个过程阶段缺陷植入和移除

开发过程的每个阶段都可能植入缺陷，在大规模的开发项目中，在代码集成之前的阶段，即需求分析、设计、编码实现阶段，可以通过评审手段来移除缺陷，代码集成之后通过测试来进行移除缺陷。在测量每一阶段缺陷移除效率时，需要考虑的问题有两点。第一点是早期阶段的评审发生时，后期阶段植入的缺陷还没有产生；第二点是每个阶段发现的缺陷有可能被不正确地修复，这些缺陷不能算作该阶段移除的缺陷。图 1 描述了每个阶段缺陷植入和移除的情况。

从图中可以看出，每个阶段的缺陷移除效率可以定义为[1]：

移除的缺陷(该步骤中)
(每个步骤开始存在的缺陷 + 该步骤中植入的缺陷) × 100%

为了得到完整的操作定义,需要提出如何测量阶段缺陷移除效率和全面缺陷移除效率。可以使用"缺陷来源/发现处"矩阵对缺陷的植入来源和发现阶段进行交叉排列来计算缺陷移除效率。在每一个阶段的缺陷移除活动中确定发现缺陷的来源(植入阶段),然后将缺陷的来源和发现处填入矩阵中。图 2 给出一个具体的矩阵排列的例子。

发现处 \ 缺陷来源	需求	高层设计	低层设计	编码	单元测试	组件测试	系统测试	应用领域	总计
需求	—								—
高层设计	39	661							700
低层设计	4	39	660						703
编码	10	26	113	929					1078
单元测试	19	41	40	220	2				322
组件测试	18	37	59	259	—	4			377
系统测试	5	6	22	70	—	—	1		104
应用领域	8	15	14	34	—	—	—	1	72
总计	103	825	908	1512	2	4	1	1	3356

图 2 "缺陷来源/发现处"矩阵

	j=1	j=2	j=3	*	*	*	*	j=k	
i=1	N_{11}								N_1
i=2	*	N_{22}							N_2
i=3	*	*	N_{33}						N_3
*	*	*	*	*					
*	N_{ij}	*	*	*	*				N_i
*	*	*	*	*	*	N_{ij}			
*	*	*	*	*	*	*	*		
i=k	*	*	*	*	*	*	*	N_{kk}	Nk
	N*1	N*2	N*3	*	*	N*j	*	N*K	总计

从图 2 中可以看出,因为缺陷发现的阶段总是不能早与缺陷植入的阶段,所以"缺陷来源/发现处"矩阵是一个三角矩阵。在这个例子中,需求阶段的审查没有发现本阶段植入的缺陷,测试阶段的对角线值代表了不正确修复的缺陷数目,所有不正确的修复都在同一阶段被发现并得到正确的修复,但有些情况下,不正确修复的缺陷直到后续阶段才被发现。

基于给出的定义,可以如下计算每个阶段以及全面的缺陷移除效率。

高层设计(I0)审查效率 - IE(I0)

移除的缺陷 -700; 阶段开始时存在的缺陷 -103;该阶段植入的缺陷 -825

$$IE(I0) = \frac{700}{103+825} \times 100\% = 75\%$$

编码(I2)审查效率 - IE(I2)

移除的缺陷 -1078; 阶段开始时存在的缺陷 -(103 +825 +908 -700 -703) =433;该阶段植入的缺陷 -1512

$$IE(I2) = \frac{1078}{433+1512} \times 100\% = 68\%$$

过程全面缺陷移除效率 - DRE

$$DRE = (1-\frac{72}{3356}) \times 100\% = 97.9\%$$

同样可得到该例子中各阶段和全面缺陷移除效率为如表 1 所示。

通过以上的例子,可以得到基于"缺陷来源/发现处"矩阵的缺陷移除效率操作定义[1]。

令 i =1,2,…,k 表示软件生命周期阶段的审查或测试类型;令 j =1,2,…,k 表示软件生命周期的阶段。

对角线上(N_{ij}，当 i=j)包含植入和在同一阶段发现的缺陷数目;对角线以下(N_{ij}，当 i > j)的单元格包含来自早期开发阶段而后来才被发现的缺陷。对角线以上的单元格是空的,因为早期的阶段不可能发现后来阶段植入的缺陷。矩阵的行(N_i)表示移除的缺陷,列(Nj)表示不同来源的缺陷。

阶段缺陷移除效率(PDREi)可以变成阶段审查效率(IE(i))或阶段测试效率(TE(i))

$$PDRE_i = \frac{Ni}{\sum_{m=1}^{i} Nm - \sum_{m=1}^{i-1} Nm}$$

全面审查效率(IE)

$$IE = \frac{\sum_{i=1}^{l} Ni}{\sum_{i=1}^{k} Ni}$$ 其中 l 是审查阶段的数目

全面测试效率(TE)

$$TE = \frac{\sum_{i=l+1}^{k-1} Ni}{\sum_{i=l+1}^{k} Ni}$$ 其中 l+1, l+2,…, k-1 是测试阶段

开发过程的全面缺陷移除效率(DRE)

$$DRE = \frac{\sum_{i=1}^{k-1} Ni}{N}$$

表 1

高层设计审查效率	75%	单元测试效率	37%	全面审查缺陷移除效率	74%
低层设计审查效率	62%	组件测试效率	68%	全面测试缺陷移除效率	92%
编码审查效率	68%	系统测试效率	59%	过程全面缺陷移除效率	97.9%

4.3 缺陷移除效率测量的意义

阶段缺陷移除效率的测量和分析对软件组织进行缺陷移除过程的改进是很有帮助的。测量的结果清楚地表明了缺陷移除过程中哪些阶段需要改进,例如:在图3.2的例子中可以看出,单元测试的缺陷移除水平较低,是需要改进的缺陷移除步骤。对于同一个软件项目,可以通过"缺陷来源/发现处"矩阵的纵向版本进行比较来评估组织缺陷移除过程的改进水平。

在整个缺陷移除过程中应该重视早期的缺陷移除效率,早期的缺陷移除效率越高,传递到下一个阶段的缺陷就越少,移除缺陷的代价也越小。通常,缺陷产生和移除的时间间隔越小,证明缺陷移除过程越有效。

5 结论

使用"缺陷来源/发现处"矩阵的方法可以来记录缺陷起源和缺陷发现阶段的数据,根据缺陷移除的公式可以进行阶段缺陷移除效率、全面效率、测试效率、审查效率的测量,这些测量为项目的缺陷移除过程提供了很好的能力指标,通过在组织内的纵向比较或与工业基线的横向比较,帮助组织改进缺陷移除的过程。

参考文献

1 Stephen H. Kan, Metrics and Models in Software Quality Engineering, 北京:电子工业出版社,2004.

2 G. Gordon Schulmeyer, James I. McManus. Handbook of Software Quality Assurance, 北京:机械工业出版社,2003.

3 Leung, H. K. N., Improving defect removal effectiveness for software development. Software Maintenance and Reengineering, 1998, 21:157 – 164.

4 Pankaj Jalote, CMM in Practice Processes for Executing Software Projects at Infosys, 北京:电子工业出版社,2002.

5 Dunn, R. H., "The Quest for Software Reliability," Handbook of Software Quality Assurance, G. G. Schulmeyer and J. I. McManus, Eds., New York: Van Nostrand Reinhold, 1987, pp. 342 – 348.

6 Jones, C., Software Assessments, Benchmarks, and Best Practices, Boston: Addison – Wesley, 2000.