一种基于UML用例模型的软件可靠性分配方法

胡文生[1,2,3] 赵 明[3] 杨剑峰[2] 贾国荣[1]

(贵州商业高等专科学校计算机科学与技术系 贵阳 550004)[1]

(贵州大学计算机科学与信息学院 贵州 550025)[2] (贵州省可靠性工程中心 贵州 550025)[3]

摘 要 基于统一建模语言(UML)用例模型描述软件系统需求方面的特点;提出了以UML用例图中所包含的各个用例的使用频率作为衡量该用例的重要程度的依据,从而确定每个用例的重要度的权重因子。根据每个用例的权重因子来确定每个用例该承担的可靠性指标任务,实现软件可靠性指标在各个用例之间进行合理的分配。这种分配方案对保证软件系统的可靠性目标具有重大的现实意义,为软件后续开发、测试、维护提供可以参考的依据。

关键词 UML,用例,可靠性,分配

中图法分类号 TP312 **文献标识码** A

Method of Software Reliability Allocation Based on UML Use Case Model

HU Wen-sheng[1,2,3] ZHAO Ming[3] YANG Jian-feng[2] JIA Guo-rong[1]

(College of Computer Science and Technology, Commercial College of Guizhou, Guiyang 550004, China)[1]

(College of Computer Science & Information, Guizhou University, Guiyang 550025, China)[2]

(Reliability Engineering Center of Guizhou, Guiyang 550025, China)[3]

Abstract We introduced a methodology that starts with the analysis of the UML use case diagram for software reliability allocation with the probability of executing each use cases. We will utilize the probability of executing each use cases to measure the importance of the use case, then to determine weight factor of each use case. According to the weight factor of each use case to determine the reliability index undertaken by each use case, system architect can reasonable assign reliability index of system to each use case. This methodology based on use case can ensure the target of software reliability and provide reference for follow-up development and software test.

Keywords UML, Use case, Reliability, Allocation

1 介绍

软件质量的一个重要度量指标就是可靠性,其一般是由用户在软件开发之前提出可靠性要求,系统分析师根据用户的要求在软件需求规格说明书中明确指出的软件可靠性目标,该目标是软件可靠性分析、设计、评估、鉴定、验收的依据。为了达到需求规格说明书中所规定的软件可靠性目标,软件开发人员在开发过程中必须充分理解软件可靠性要求,有目的、有组织、有计划地将软件可靠性指标采用合适的方式分配给各配置项、软部件直至软件单元和相关人员。这就涉及到软件可靠性分配问题。所谓软件的可靠性分配,就是指按照软件需求规格说明书中所规定的可靠性指标将其合理地分配到组成软件系统的每一个单元。可靠性分配的目的就是要落实软件系统的可靠性指标、明确系统各组成单元的可靠性要求、明确软件开发人员在可靠性方面所应该承担的责任、为系统设计或系统测试提供依据[1]。目前,软件可靠性分配主要建立在软件及软件开发过程中固有特征的基础上,充分考虑软件的重要度、复杂性、运行时间等各方面的因素,在软件需求分析和软件设计阶段不断反复迭代,最终实现软件可靠性目标。软件可靠性分配可以从定性和定量的角度进行考虑,但是在软件需求分析和设计阶段,定量分析所需要的数据是很难获取的,所以一般都是采用定性的软件可靠性分配,主要有面向程序结构的重要度、复杂度等软件可靠性定性分配方法。由于软件的复杂度的度量涉及到对各个软件模块复杂度的评价,有一定的难度,因此目前比较流行的软件可靠性分配方法是基于重要度的可靠性分配方法。基于重要度的分配方法基本原则是根据构成软件的各个模块失效时对系统造成的影响程度来分配软件系统的可靠性,重要等级越高,分配的可靠性指标值就越高。而UML模型能够从静态和动态的角度全方位地描述整个软件系统,其中UML用例图反映了系统需要完成的全部功能,依据UML用例图中的各个用例的使用频率能够清楚地表明各个用例的重要程度。本文是从UML用例图出发,研究用例图中各个用例的出现频率来确定各个用例的重要程度,进而分配可靠性指标。该方法相对

本文受贵阳市科技局项目([2010]筑科Ⅰ合同字第1-61号)资助。

胡文生(1969-),男,博士生,讲师,主要研究方向为软件可靠性,E-mail:hwsgzsz@yahoo.com.cn;**赵 明**(1962-),男,博士,教授,博士生导师,主要研究方向为软件、硬件可靠性、应用统计;**杨剑峰**(1986-),男,博士生,主要研究方向为软件可靠性;**贾国荣**(1958-),男,教授,主要研究方向为电子商务。

于其它可靠性分配方法来说，实现起来简单，而且更容易理解。

本文第1节主要介绍一些背景知识；第2节具体介绍基于UML模型的软件可靠性分配策略；第3节主要介绍该策略分析的具体应用过程；最后总结和明确未来的研究方向。

2 基于UML模型的软件可靠性分配策略

2.1 UML用例模型介绍

UML模型能从动态和静态角度全方位地描述一个软件系统，其中UML用例模型属于UML静态模型，它由多个用例图组成。一个用例图由系统边界、参与者、用例以及它们之间的关系组成。图1所示就是一个简单的用例图。

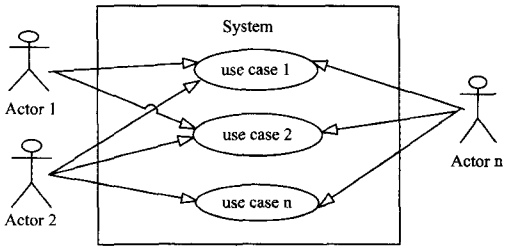

图1 UML用例图

用例图可以有多个参与者和多个用例，一个参与者可以与多个用例建立连接，表示该参与者是这些用例的主要执行者。参与者可以是人，也可以是其他软件系统或硬件系统。用例与用例之间存在不同的关系，主要有继承、关联、包含等关系[2]。本文着重考虑用户与用例之间的相互交互情况来确定用例的重要度，所以用例之间的关系不是本文讨论的重点。在用例图中每个用例表示系统的一个需求，一个正确的用例图能涵盖软件需求规格说明书中所描述的所有需求，而需求是软件的源动力和最终的目标，它贯穿于软件生命周期的各个阶段，所以描述软件需求为任务的UML用例图就是其他UML模型图(顺序图、交互图、类图、包图、部署图、对象图、状态图等)的基础。

2.2 确定各用例出现的频率

在UML模型中，一般用用例模型来表述用户的需求，一个好的用例模型能够完整地表述出用户的所有需求，其中用例模型中的每个用例就代表用户的一个需求。所以敏捷开发过程由原来以需求为基础进行迭代，演变为以UML用例模型中的用例为基础的不断反复、渐增的过程。UML的用例模型一般是由3部分组成：参与者、用例、系统边界。为了确定用例模型中每个用例出现的频率，采用适当的标识符来标识UML用例模型中的有关元素。用 p_i 表示第 i 类用户出现的频率，p_{ij} 表示第 i 类用户使用UML用例模型中的第 j 个用例的频率。于是假设某个用例模型中有 N 个用户类型，某个用例 x 被用户使用的频率为[3]：

$$p(x)=\sum_{i=1}^{n} p_i * p_{ij} \tag{1}$$

2.3 确定各用例重要度

根据各用例使用的频率来确定用例的重要度，一般认为频率越高的用例对用户来说是越关键的功能，它对应的重要度就越高。根据用例的使用频率，经过专家的评定为每个用例指定一个相应的重要度权值并按照权值对它们进行排序。确定重要度的权值有两种方式，一种是以软件的可靠度作为软件系统可靠性的目标值，在这种方式下把使用频率最高的用例的重要权值确定为最大值，然后依次确定其他用例的权重并按照从大到小的顺序进行排序；另一种方式是以软件失效率作为软件系统可靠性的目标值，把出现频率高的用例的重要权值确定为最小值，即重要权重值越小，表明该用例的重要程度越高，然后依次确定其他用例的权重并按照从小到大的顺序进行排序。

2.4 各个用例分配可靠性指标

首先确定整个系统的软件可靠性目标值为 R；其次确定系统用例总数，比如假设系统总共有 n 个用例，确定各个用例的重要度权值 U_k(其中 $k=1,2,\cdots,n$)；最后计算出每个用例所分配的可靠性指标值 R_k 为：

$$R_k=\frac{U_k}{\sum_{i=1}^{n} U_i} * R \tag{2}$$

若以软件系统的总失效率 f 为目标值，则各个用例分配的失效率的指标值 f_k 为：

$$f_k=\frac{U_k}{\sum_{i=1}^{n} U_i} * f \tag{3}$$

3 该可靠性分配策略具体应用

在这一节中我们重点讨论一个具体的ATM应用实例，研究该如何依据ATM用例图来实现软件可靠性分配。在UML用例图加上相应的标注，主要标注的是各个用户类型所占整个用户群体的概率，以及各个用户使用用例的概率。这个系统的用例图如图2所示。

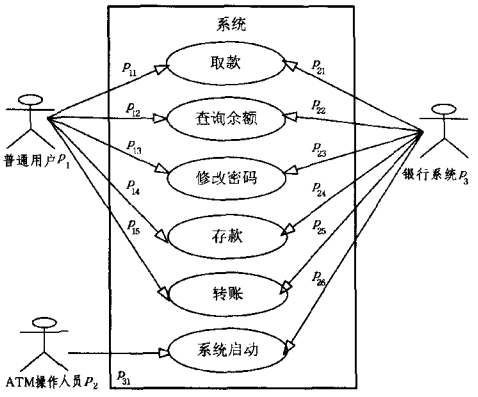

图2 加标注的ATM机用例图

做如下假设：

(1)各用户类型出现的概率分别为：

$p1=0.6, p2=0.39, p3=0.01$

(2)普通用户使用各个用例的概率为：

$p11=0.6, p12=0.2, p13=0.01, p14=0.1, p15=0.09$

(3)银行系统使用各个用例的概率为：

$p21=0.6, p22=0.2, p23=0.0199, p24=0.1, p25=0.08, p26=0.0001$

(4)ATM操作人员使用各个用例的概率为：

$p31=1$

于是根据式(1)可以计算出各个用例的出现频率：

P(取款)$=p1*p11+p2*p21=0.594$

P(查询余额)$=p1*p12+p2*p22=0.198$

P(修改密码)$=p1*p13+p2*p23=0.013761$

P(存款)=$p1*p14+p2*p24$=0.099

P(转账)=$p1*p15+p2*p25$=0.0852

P(系统启动)=$p2*p26+p3*p31$=0.010039

根据整个系统的用例使用概率，以软件失效率作为目标值并经专家评定给它们分配重要度权值，按照从小到大的顺序对用例进行排序，计算方式如下：

(1)记 w_i 表示第 i 个用例的重要度权值。p_i 表示第 i 个用例的出现频率。$ratio=p_1-p_2$。

按照软件系统用例出现的频率从大到小排列。设出现频率最高的用例的重要度权重值为 w_1(这里假设为 $w_1=1$)，出现频率次高的用例的重要度权重值为 w_2(这里假设为 $w_2=3$)。

(2)用例之间的相差重要度权值为：

$$\Delta w_i=\frac{(w_2-w_1)*(p_{i-1}-p_i)}{ratio},i=3,\cdots,n$$

(3)于是从第 3 个用例开始，各个用例的权重值为：

$w_i=w_{i-1}+\Delta w_i$

具体情况见表 1。

表 1　各个用例权重的确定

用例名	概率	权值
取款	0.594	1
查询余额	0.198	3
存款	0.099	3.5
转账	0.0852	3.57
修改密码	0.013761	3.92
系统启动	0.010039	3.94
总计		18.93

根据表 1 进行可靠性分配计算。假设该系统要求可靠性目标为失效率为 0.001 失效数/小时，则可以根据式(3)计算出各个用例对应的失效率分配值。

(1)取款用例的失效率分配值为：

$f1=(1/18.93)*0.001=0.0000528$

(2)查询余额用例的失效率分配值为：

$f2=(3/18.93)*0.001=0.0001584$

(3)存款用例的失效率分配值为：

$f3=(3.5/18.93)*0.001=0.0001848$

(4)转账用例的失效率分配值为：

$f4=(3.57/18.93)*0.001=0.000188496$

(5)修改密码用例的失效率分配值为：

$f5=(3.92/18.93)*0.001=0.000206976$

(6)系统启动用例的失效率分配值为：

$f6=(3.94/18.93)*0.001=0.000208032$

这表明要想满足整个软件系统的失效率为 0.001 失效数/小时的目标，必须要求取款用例的失效率控制在 0.0000528 失效数/小时范围内。其他用例都有类似要求，这种结果对软件测试人员具有很好的参考价值。

结束语　UML 是目前最流行的软件建模语言。一个正确的 UML 用例模型能如实地反映软件系统需求规格说明中所规定的各种需求。利用用例的出现频率来计算各个用例所应该承担的可靠性指标，可为后续阶段的软件开发和软件测试提供很好的参考价值。但是这种方法也有一定的缺陷，主要是确定重要度权值需要依赖专家的评审，而对具体软件开发成员该承担多少可靠性指标任务无法衡量。所以我们将利用 UML 用例图、顺序图、交互图、状态图、类图等模型图，依据 UML 模型图所具有的层次结构特点对软件可靠性指标逐级分解，从而确定每个类该承担多少可靠性指标任务，最终对每个软件开发成员在可靠性方面所应该承担的责任进行很明确的规定，以真正体现“软件的可靠性是设计出来的”精神。

参 考 文 献

[1] 孙志安，裴晓黎，宋昕，等. 软件可靠性工程[M]. 北京：北京航空航天大学出版社，2009，3：160-175

[2] Eriksson H-E, Penker M. UML Toolkit[M]. Publishing House of Electronics Industry, 2004, 10: 150-230

[3] Singh H, Cortellessa V, Cukic B, et al. A bayesian approach to reliability prediction and assessment of component based systems[C]// Proc. of 12th International Symposium on Software Reliability Engineering(ISSRE'01). 2001

[4] Corteliessa V, Singh H, Cukic B. Early reliability assessment of UML based software models[C]// WOSP'02. Rome, Italy, July 2002: 24-26

[5] Khalaj M, Makui A, Tavakkoli-Moghaddam R, et al. Systematic Approach to Calculate Risk and Reliability in Uncertainty Condition[J]. Journal of Basic and Applied Scientific Research, 2012, 2(1): 573-582

[6] Hu Weng-sheng, Deng Zhou-hui, Hong Yi. A Method of FTA Base on UML Use Case Diagram[C]// ICRMS' 2011. June 2011: 12-15

[7] Fan Lin-bo, Ma Zhi-gang. Tendency Analysis of Software Reliability Engineering[C]// ICRMS'2011. June 2011: 12-15

(上接第 443 页)

[3] 多核系列教材编写组. 多核程序设计[M]. 北京：清华大学出版社，2007(9)：201-203

[4] Kunz T. The influence of different workload description on a heuristic load balance scheme[J]. IEEE Trans on Software Engineering. 1991, 17(7): 725-730

[5] 王力生，毛昀波. 多处理机系统的负载平衡模型设计[J]. 单片机与嵌入式系统应用，2008(4)：10-13

[6] 胡亮，徐高潮，鞠九滨. 一个基于收益与开销的作业选择策略[J]. 软件学报，1998，9(4)：280-283

[7] 方娟，蒲江，张欣. 片上多核处理器共享 Cache 划分的公平性研究[J]. 计算机工程与设计，2010，31(15)：3413-3415，3517

[8] Stone HS, Turek J, Wolf J L. Optimal Partitioning of Cache Memory[J]. IEEE Trans on Computer, 1992, 41(9): 1054-1068

[9] 所光，杨学军. 多核处理机系统 Cache 管理技术研究现状[J]. 计算机工程与科学，2010，32(7)：65-68

[10] Tam D, Azimim L, Stamm M. Managing shared L2 Caches on multi-core systems in software[C]// Workshop on the Interaction between Operating Systems and Computer Architecture. 2007

[11] Kim S, Chandra D, Solihin Y. Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture[C]// Proceedings of the 13th International Conference on Parallel Architecture and Compilation Techniques. 2004: 111-122

[12] Iyer R. CQoS: a Framework for Enabling QoS in Shared Caches of CMP Platforms[C]// Proc of the 18th Annual Int' l Conf on Supercomputing. 2004: 257-266

[13] Moreto M. FlexDCP: a QoS framework for CMP architectures. ACM SIGOPS Operating Systems[J]. 2009, 43(2): 86-95