基于用例的软件复杂度估算及应用

王 悠[1,2],罗燕京[1],易福华[1],房 芳[1]
(1. 北京航空航天大学 计算机学院 软件工程研究所,北京 100083;
2. 中国民航飞行学院 航空工程学院,四川 广汉 618307)

摘 要:用例驱动是RUP开发过程的要素之一。研究基于用例的软件复杂度,对于项目规模估算、进度控制和度量、评估都具有积极的意义。讨论了在用例驱动的软件开发过程中如何获取、量化用例层次上的软件复杂度的方法,概括了其研究的意义,并根据它对软件可靠性相关指标参数的影响,探讨基于用例的软件可靠性度量分析方法的改进。

关键词:UML;用例复杂度;用例关系;缺陷指数
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2007)07-0196-04

Estimating Software Complexity Based on Use Cases and Its Application

WANG You[1,2], LUO Yan-jing[1], YI Fu-hua[1], FANG Fang[1]
(1. Software Eng. Institute, School of Computer, Beihang University, Beijing 100083, China;
2. Aviation Eng. School, Civil Aviation Flight University of China, Guanghan 618307, China)

Abstract: Use case-driven method is one of the elements in rational unified process. Researching on software complexity based on use cases acts an important part in project size estimate, scheduling, measurement and assessment. A method and its significance for accounting and acquiring the software complexity during the use case-driven development process are discussed in detail in this paper. It also focuses on how to improve the measurement of the software reliability based on use cases.

Key words: UML; complexity of use cases; use case relationship; defect index

0 引 言

如何有效地估算软件开发活动的工作量及产品复杂度,一直都是开发过程中的难题。而软件开发过程中,主要通过软件度量方法来评估项目的复杂度,然后根据这些复杂度进行人员安排、进度控制以及度量评估等各项管理活动。然而传统的软件估计方法包括:功能点分析和源代码行分析,及基于数组实现的软件尺度评估,这些方法只是对软件开发的某一阶段的工作量进行了估计,只能把握某一阶段的进度,而不能够连续性地控制整个软件的开发进度。如源代码行数估计只能针对编码阶段的控制,而用例分析技术和UML的广泛使用,使人们有可能在项目的需求获取和设计阶段对整个项目复杂度做出较准确的估算,为今后的开发管理活动提供参考。

收稿日期:2006-09-21

作者简介:王 悠(1978-),女,四川广汉人,硕士研究生,讲师,研究方向为UML、面向方面开发技术、软件项目管理;罗燕京,副教授,硕士生导师,研究方向为UML、RUP、软件工程和软件工程环境。

1 用例分析技术

用例分析的思想已经非常成熟,特别是用例与工业标准的建模语言UML[1]的结合,使得用例分析在新的软件开发统一过程中已经占据决定性的地位。目前最流行的软件过程之一Rational Unified Process(RUP)是一个使用生命周期迭代法的软件开发过程,而它的要素之一就是用例驱动[2]。RUP的四个主要阶段,即初始、细化、构造和交付阶段都含有用例的应用。正是由于用例在开发过程中的重要地位,因此,与用例相关的估算和度量,对整个项目开发的事前估计性、预防性、评估性等多方面都起到积极的作用。

用例的表达,通常采用UML作为建模语言。所以在考虑进行需求和设计的度量时选用其最常用的UML语言作为中间产品的表述形式。UML语言用用例图、类图、包图、状态图、活动图、顺序图、合作图、构件图和配置图进行需求、设计和系统分布的描述。参照常用流程,定义一种表示方法如图1所示。

2 基于用例的软件复杂度估算

基于用例的需求分析和设计度量体系是以整个项

目为对象,包括用例数、执行者数并针对它们的属性提出的。针对 UML 语言提出的相关用例的度量指标,逐一进行分析。按照图 1 中用例分析技术[3]的基本方法和步骤,逐步讨论相关复杂度的估算。

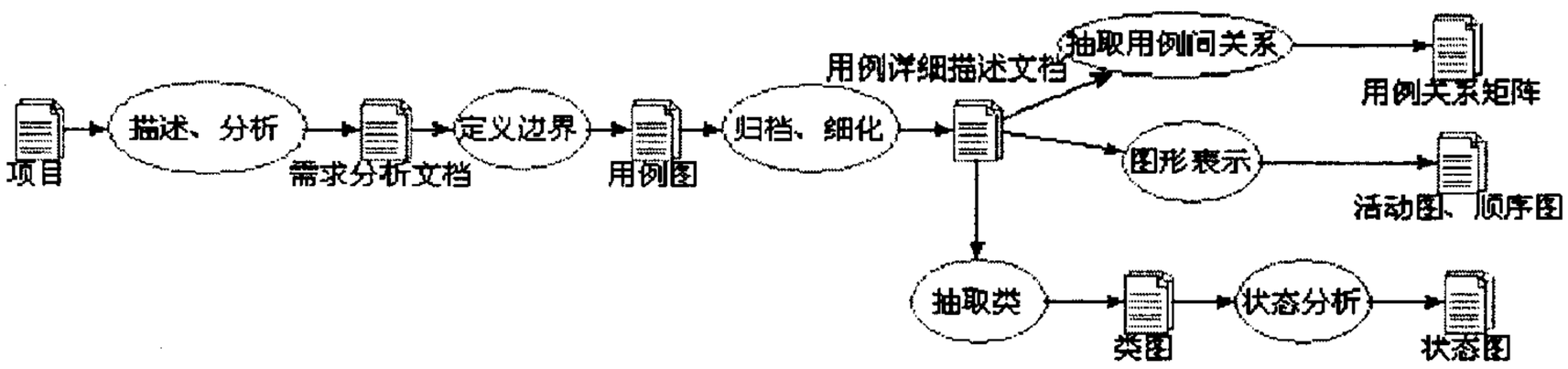

图 1　用例获取分析活动图

2.1　项目描述及风险分析

主要工作是把要做的事情陈述出来,以正确把握系统的功能需求,并寻找影响项目成功和可能导致失败的各种因素,从而避免这些风险。在这个阶段中,可以获取的度量维度是项目维的[4],包括下面的几个方面:

指标 1:用例数目 NOU 和执行者数目 NOA。

指标定义:这两个数目是指在一个项目中的用例的数目(Number Of Usecases),所有的执行者的数目(Number Of Actors)。

应用意义:一般而言,这二者的数量越多,系统越大,这些指标与系统的大小近似成正比。仅从这几个指标上不能准确比较系统的情况,需要对系统更详细的指标进行分析度量。

2.2　确定系统边界

确定系统边界意味着找出系统之中有什么(用例),系统之外有什么(参与者),通过捕获参与者确定用例和描述用例获取信息。从这个过程开始,将获取更详细的用例信息,度量维度涉及用例和执行者维中的每个用例(执行者)的属性。

指标 2:用例相关执行者数 ARU(Actors Relative to the Usecases)和执行者相关用例数 URA(Usecases Relative to the Actor):是指与之有通信关系(Communicates relationship)的执行者(用例)的个数。

ARU 与 RAU 的应用意义为:

(1)与某个用例相关的执行者个数反映的是某个用例的重要程度,如果这个数字很大,说明这个用例在系统中影响较大,然而如果一个用例相关的执行者个数过多,就要考虑用例的获取是否恰当,是否有必要将该用例再划分为几个用例。

(2)与用例相关的执行者说明用例获取的来源,对重要的用例应多和相关角色(由工作分工确定)交流。

(3)同理,与某个执行者相关的用例个数,从侧面反映了该执行者的重要程度,优秀的用例应该是唯一(unique)的,如果执行者相关用例数过大,应考察这些用例有无重复,或者重新评估执行者的划分是否正确。

指标 3:消息维大小 MS(Message Size),相对消息维大小 RMS(Relative Message Size):

MS=某用例(执行者)中传递的消息数目;RMS $= MS_i / \sum MS_i$

应用意义:

(1) $\sum MS_i$ 可以估计项目的大小。

(2)通过 MS 可以估计用例划分的合理情况。当这个指标数据过大时,该用例就应该进行进一步审核,以确定其划分的合理性,反之,数据过小,同样也要审核。

(3)RMS 反映了某个用例就消息意义上在整个项目中所占的比值,它反映了该用例在系统中所处的地位。

2.3　用例归档、细化

为用例创建详细的文档,给项目的需求增添细节,这些细节告诉人们需要完成哪些步骤才能实现这个用例的功能。在细化事件流的过程中,必须清楚用例之间的相互关系,才能分清主次轻重,合理安排工作。用例之间的关系有三种,即包含、扩展和继承关系。利用关系矩阵,能清楚、简便地记录用例的相互关系,如表 1 所示。其中 Inc_{xy} 表示用例 x 包含用例 y;Ext_{xy} 表示用例 x 扩展了用例 y;Inh_{xy} 表示用例 x 继承了用例 y。

表 1　用例关系矩阵

	UC1	UC2	UC3	UC4	UC5	UC6
UC1	0	Inc	Inc	0	0	Inh
UC2	0	0	0	Ext	0	0
UC3	0	Inc	0	Ext	0	0
UC4	0	0	0	0	0	0
UC5	0	Inc	Inc	0	0	0
UC6	0	0	0	0	Inh	0

指标 4:用例加权关系数:Weighted Methods per Relation(WMR),相对加权关系数 WMR′。

从关系矩阵中,按行叠加各单元值获取用例加权关系数 WMR 及按列叠加相对加权关系数 WMR′。如公式(1):

$$WMR_x = \sum_{y=1}^{n} W_{xy} \quad WMR'_y = \sum_{x=1}^{n} W_{xy} \qquad (1)$$

可以明确的是,由关系矩阵所获得的 WMR 能够表明一个用例与其它用例间关系的复杂程度,WMR 值越大,说明该用例对另一用例的敏感度越大,维护不易,在设计和实现的时候应给予更高的关注,评审和测

试力度应相应加大。如果 WMR 过大,则应考虑架构的重新设计。而 WMR′很好地表示了它对其它用例的重要度及影响,在设计和修改这些用例的时候,必须谨慎,因为修改这个用例,很可能会影响到其它相关用例的使用。按照实践经验,三种关系复杂度的权值 Inc、Ext、Inh 分别对应值 0.5、1、2。例如按表 1 所示,可以得到:

$WMR_{UC1} = Inc + Inc + Inh \times Inh = 0.5 + 0.5 + 2 \times 2 = 5$①

$WMR'_{UC2} = Inc + Inc + Inc = 1.5$

$WMR'_{UC4} = Ext + Ext = 2$

2.4 用活动图或序列图表示用例

活动图针对每个用例进一步描述用例包含的步骤,活动图将用例主事件流和备选事件流之间的关系用图形的方式表示,因此,对用例复杂度的计算由判断主事件流和备选事件流的个数、关系,进而转化成对活动图相关元素的分析。根据活动图的特点,提出下面的公式。

指标 5:用例复杂度 CU(Complexity of Usecases):复杂度是由活动图中所使用的判断结点、同步结构、循环结构等特殊结构决定的。其定义公式(2)为:

$$Complexity(CU) = \sum_{i=1}^{k} C_i + \sum_{i=1}^{n} W_i + \sum_{i=1}^{m} D_i \times T_i + \sum_{i=1}^{l} S_i \quad (2)$$

其中,k 为循环个数,C_i 为第 i 个循环结构的权值;n 为顶层普通结点(活动结点或状态结点)的个数,其中第 i 个普通结点的权值为 W_i;m 为顶层判断结点的个数,其中第 i 个判断结点的分叉数为 T_i,权值为 D_i;l 为顶层并发结构的个数,第 i 个并发结构的权值为 S_i。对于一般结点,如果其存在嵌套子图,则以这个子图为度量对象,转入子图类似计算,否则,其内部 Action 可分为四种:On entry,On exit,Do,On event,若令它们的权值为 W_a,前三种 Action 个数总共为 r,On event 的事件发生概率为 P_i,其 Action 个数为 s,则有下面公式(3):

$$W_i(或\ S_i) = r \times W_a + \sum_{i=1}^{s} P_i \times s \times W_a \quad (3)$$

通常,判断结点及其分叉数越多则需求结构越复杂;循环越多则需求结构越复杂;泳道的出现只是改变了图形的表示方式,对需求复杂性没有影响;同步事件的出现也会使待开发系统的复杂性增加;一般状态结点和活动结点的出现会导致系统的规模和复杂性增加。

对于用例复杂度的估算其应用意义在于:用例复杂度基本反应了该用例的工作量和难度,可以用来估计用例优先级(核心的、重要的、辅助的)和完成这个用例所需要投入的工作量。并且,获取的用例复杂度的相关数据,也可应用到相应的用例质量评估分析之中。

文中提出的度量理论在以往度量理论基础上结合 UML 的特点形成。而有关类维的度量,已经有相当成熟的面向对象度量方法可以使用,例如典型的面向对象度量 C&K 方法。因此文中不再对面向类的度量做详细介绍。

3 应 用

软件度量学的最终目的是服务于软件质量控制与评价[5],软件过程改进必须以软件度量为基础。软件度量学经历了面向过程软件度量和面向对象软件度量的阶段。随着用例分析技术的广泛使用,如何合理利用用例的度量数据做出软件质量评价,指导相应的过程改进,将更深入地进行研究。根据用例复杂度的意义,通过研究与实践,参考 IEEE Guide 982.2[6]中相关软件可靠性分析的指标,对此进行研究和探讨,使之更具有现实意义和指导作用。

3.1 缺陷密度 DD(Defect Density)

测量计算每个用例中的缺陷数,相关原始数据记录包括内容如表 2 所示。

表 2 缺陷密度记录表

符号	i	N_i	C_i	S_i	U	D_i
含义	缺陷编号	缺陷个数	缺陷类型	严重程度	用例编号	发现日期

平均缺陷密度 ADD(Average Defect Density):

$$ADD = \sum N_i / \sum N_u \quad (4)$$

由于用例的复杂度不同,单纯的以缺陷个数或平均缺陷密度来判定该用例相关质量指数是较片面的,因此引入了用例缺陷密度 DDU(Defect Density of the Usecases):

$$DDU = \sum_i N_{ui} / CU_u \quad (5)$$

该指标可以更好地应用在迭代开发的过程中,具有更合理的参考价值。将此与期望缺陷密度作比较以预测遗留的缺陷数;也可以根据已定义的重要度级别目标,确定是否进行足够的评审和测试;根据它建立标准的用例缺陷密度,为以后的预测和比较奠定基础。

3.2 用例缺陷指数 DI(Defect Index)

用例的缺陷指数是在缺陷密度的基础上,更全面地对用例进行分析评判,它提供了一个在开发周期中持续记录软件被正确开发的程度的指数。使用时通过每个阶段赋予不同的权重来进行计算,相应记录的原

①在计算继承关系时应递归向上,直到顶层的父用例,即继承树中不是任何用例子用例的用例,按层乘积。

始数据如表 3 所示。

表 3　用例缺陷记录表

符号	i	U	N_{ui}	$S_{ui}M_{ui}T_{ui}$	W
含义	阶段编号	用例编号	某用例阶段缺陷总数个数	分别表示严重、中等、轻度缺陷总数	缺陷权重

说明:W_s = 严重缺陷权重(缺省为 10);W_m = 中等缺陷权重(缺省为 3);W_t = 轻度缺陷权重(缺省为 1)。

在每个软件开发阶段,根据缺陷严重性和数量获取公式(6):

$$P_i = W_s \times (S_{ui}/N_{ui}) + W_m \times (M_{ui}/N_{ui}) + W_t \times (T_{ui}/N_{ui}) \tag{6}$$

用例缺陷指数(DI) 通过把软件开发各阶段的 P_i 通过下面公式(7) 计算得到:

$$\mathrm{DI} = \frac{\sum_{i=1}^{n} i \times P_i}{\mathrm{CU}} = (P_1 + 2P_2 + 3P_3 + \cdots)/\mathrm{CU} \tag{7}$$

软件生命周期各阶段也被标以一个权重,软件开发的进展越多,例如阶段 2 或阶段 3,那么赋予的权重越大(也就是 2 或 3),最后把结果按该用例的复杂度(CU)规格化。该指标可以将从以往的项目或用例得到的数据用作比较的基线数字,并将为下次迭代更新基线。可以确定的是,相关用例的缺陷密度和缺陷指数在用例的审查和评估中,能提供客观有效的参考数据。

4　总　结

对用例相关的度量,由于度量对象的特殊性导致该研究具有独特的意义:

(1)预防性:与以往的一些研究(比如针对源代码的度量)不同,这里是针对需求分析模型和设计模型进行的度量。将问题发现并杜绝在软件开发的早期阶段,会大大减少由于错误或不合理而导致的花费。

(2)事前估计性:进行项目的估计和预算是软件开发较为关键的一环。所提出的度量指标能够对项目进行有效的估计,从而能够帮助开发部门合理调度资源,做好软件开发计划。

(3)评估性:所做的工作为比较开发过程提供了一定的量化依据。对于软件过程改进开始时基线的识别、改进进行到一定阶段后改进效果的衡量,提供切实可行的操作方法。

(4)实用性:研究是针对目前的主流开发过程 RUP——以用例驱动为中心,涉及的建模语言 UML 也是在软件工程界中占据着主导地位,因此对此的研究具有积极的意义。

可以预见到的是:随着用例技术的更普遍使用,基于用例的相关度量分析将更加深入而广泛。

参考文献:

[1] Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual[M]. [s. l.]: Addison Wesley Longman, Inc,1999.

[2] Jacobson I. Object - Oriented Software engineering - A Use Case Driven Approach[M]. Harlow, England: Addison - Wesley,1992.

[3] Schneider G. 用例分析技术[M]. 第 2 版. 北京:中信出版社,2002.

[4] 潘秋菱. 基于过程和度量的软件质量管理方法研究[D]. 合肥:合肥工业大学,2002.

[5] Schulmeyer G G, McManus J I. Handbook Software Quality Assurance[M]. 北京:机械工业出版社,2003.

[6] IEEE Standard for a Software Quality Metrics Methodology, IEEE Std[S]. 1989.

(上接第 162 页)

防御措施已难以应付。文中提出的基于历史信任数据的 DDOS 防御模型,引入了信任数据库作为辅助。在网络流量发生异常时,启动源地址检测,对进入边界路由器的异常包通过拥塞控制算法和一定的信任机制进行过滤,丢弃掉大部分具有威胁的包。但对于所丢弃的包是否可以导入缓冲区并结合源地址追踪技术,以进一步提高该防御模型防御 DDOS 攻击的能力还有待于进一步探讨。

参考文献:

[1] 黄志洪. 现代计算机信息安全技术[M]. 北京:冶金工业出版社,2004:234 - 244.

[2] 朱良根,张玉清,雷振甲. DOS 攻击及其防范[J]. 计算机应用研究,2004(7):82 - 84.

[3] 薛立军. 分布式拒绝服务攻击检测与防护[D]. 西安:西安电子科技大学,2003.

[4] 罗光春. 入侵检测若干关键技术与 DDOS 攻击研究[D]. 西安:西安电子科技大学,2003.

[5] CipherTrust. CipherTrust's Zombie Stats[EB/OL]. 2006 - 07 - 15. http://www. ciphertrust. com/resources/statistics/zombie. php.

[6] 严蔚敏. 数据结构(C 语言版)[M]. 北京:清华大学出版社,2002.