

敏捷开发过程中的迭代策略分析

胡文生^{1,2,3}, 赵 明³, 杨剑峰^{1,3}, 贾国荣²

(1 贵州大学 计算机科学与信息学院, 贵州 贵阳 550025; 2 贵州商业高等专科学校 计算机科学与技术系, 贵州 贵阳 550005; 3 贵州省可靠性工程中心, 贵州 贵阳 550025)

摘 要: 该文基于统一建模语言(UML)用例模型在描述系统的功能方面的特点结合敏捷开发过程中的迭代顺序, 提出了以 UML 用例图所包含的各个用例的风险和价值组合进行高低排序, 从而决定敏捷开发的迭代顺序. 这样做的一个明显好处就是能保证对高风险高价值的用例优先开发, 使其在随后的各次迭代中都得到测试, 从而提高该用例的可靠性, 进而提高整个系统的可靠性.

关键词: UML; 用例; 迭代; 风险; 价值

中图分类号: TP311.11

文献标识码: A

文章编号: 1000-7180(2012)05-0165-05

Iterative Strategy Analyze During Agile Development

HU Wen-sheng^{1,2,3}, ZHAO ming³, YANG Jian-feng^{1,3}, JIA Guo-rong²

(1 College of Computer Science & Information, Guizhou University, Guiyang 550025, China;

2 Commercial College of Guizhou, Guiyang 550005, China;

3 Reliability Engineering Center of Guizhou, Guiyang 550025, China)

Abstract: In this paper, We used characteristic of the Unified Modeling Language (UML) use case model in describing the function of the system's and the iteration sequence of agile development process to obtain a new iteration sequence based on the risk an value of the various use cases in UML use case diagrams. An obvious advantage of doing so is to ensure high-risk and high-value use cases will be first developed, so that the use cases will be repeatedly tested in the next iteration, thereby increasing overall system reliability.

Key words: UML; use case; iterative; risk; value

1 引言

目前软件项目的需求变化极快,从而导致软件需求的不确定性越来越高,因此软件开发组织必须具备根据需求变化及时做出调整的能力,系统分析师、系统架构师、技术开发成员与客户之间的实时沟通、及时响应客户的需求变更是保证项目成功的前提. 而传统的软件开发方法恰恰是限制系统开发与客户之间的互动,客户只是在系统开发的前期即软件需求分析阶段参与系统的开发,在系统开发后期,即软件设计、实现、测试阶段,客户完全被排除在外,在这种开发方法指导下开发出来的软件产品无法让

客户满意. 由此可见,传统的软件开发方法是导致“软件危机”的根源.

为了解决“软件危机”问题,计算机领域的专家、学者、IT 从业人员做了大量的工作. 在 20 世纪 90 年代末,极限编程、自适应软件开发、动态系统开发等一系列敏捷开发方法相继被提出. 为了既能满足客户的需求目标,又能满足客户的质量目标,在开发过程不能仅仅需要对用户需求的变化做出积极的响应,还必须把系统的安全性、可靠性等质量指标考虑进去. 为了既保证软件的质量,又保证满足客户的需求,在敏捷开发过程中有必要综合多方面因素来决定迭代顺序.

2 敏捷开发的策略分析

敏捷开发方法采用循环、迭代增量开发的方式,把用户的需求逐步添加到软件中,每完成一部分需求,就可以得到本次迭代产生的软件产品经测试后交给用户去评判,获得反馈信息,下一次迭代开发是以用户新的需求和上一次的反馈信息为基础,这样不断的循环下去,最后交付给客户的必定是用户认可的软件产品.敏捷开发迭代规划的基础一般都是以系统的功能点作为计量单位,以功能中所包含的价值高低来划分优先级,从而决定迭代的顺序.在文献[2]中提出的按照风险-价值关系确定迭代的开发顺序,如图1所示,把功能的价值和风险的关系分别映射到四个象限中.

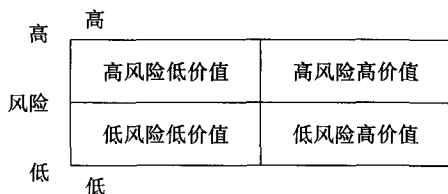


图1 软件系统功能风险价值图

从图1显示的结果可以确定迭代顺序,首先开发高价值、高风险的功能,接下来是高价值、低风险的功能,其次处理低价值、低风险的功能,最后处理低价值、低风险的功能.尽可能把高风险、低价值的功能推迟到项目之外,没必要为只有有限价值的功能冒很高的风险.把高风险高价值的功能放在敏捷开发的第一个迭代周期去实现目的是为了尽早对它采取各种纠正措施以减轻它对整个系统可靠性的影响.

在上述观点的基础上,本文提出了利用UML用例模型来确定敏捷开发过程中的迭代顺序,UML用例模型表达系统外部事物(一般为执行者或其他系统)与系统之间的交互情况及系统需要完成的各项功能,一个系统的用例模型是由若干用例图组成,而用例图是由用例、执行者、系统边界组成.在用例图中功能是以用例来表示的,每个用例指明了一个完整的功能,系统边界定义了一个系统所具有的功能,通过执行者与系统中的用例进行交互,具体描述系统是如何提供这些功能来完成相应的服务.所以对系统功能的价值和风险评估,在UML用例模型中就变成了对用例的价值和风险评估.具体的评估策略如下:

第一步:计算出UML用例模型中每个用例的

价值和风险及出现的概率.用表格形式表示出来.

第二步:根据用例的价值和风险情况把整个用例集划分成用例子集,为表示方便用下列符号表示各种集合:

U_k :表示UML用例图所包含的全部用例集合.

U_{hr} :表示UML用例图中属于高风险的用例集合.

U_{lr} :表示UML用例图中属于低风险的用例集合.

U_{hv} :表示UML用例图中属于高价值的用例集合.

U_{lv} :表示UML用例图中属于低价值的用例集合.

U_{hrhv} :表示高风险高价值的用例集.

U_{hrlv} :表示高风险低价值的用例集.

U_{lrhv} :表示低风险高价值的用例集.

U_{lrlv} :表示低风险低价值的用例集.

在上述表示的几个集合中具有如下关系:

$$U_{hr} \cap U_{lr} = \emptyset (\text{空集})$$

$$U_{hv} \cap U_{lv} = \emptyset (\text{空集})$$

$$U_{hr} \cup U_{lv} = U_k (\text{全集})$$

$$U_{hr} \cup U_{lr} = U_k (\text{全集})$$

$$U_{hrhv} = U_{hr} \cap U_{hv}$$

$$U_{hrlv} = U_{hr} \cap U_{lv}$$

$$U_{lrhv} = U_{lr} \cap U_{hv}$$

$$U_{lrlv} = U_{lr} \cap U_{lv}$$

当 $|U_k|$ 为偶数时 $|U_{hr}| = |U_{lr}| = \left\lceil \frac{|U_k|}{2} \right\rceil$;当 $|U_k|$ 为奇数时 $|U_{hr}| = \left\lceil \frac{|U_k|}{2} \right\rceil + 1, |U_{lr}| = \left\lceil \frac{|U_k|}{2} \right\rceil$.

其中 $\lceil \cdot \rceil$ 表示为集合的基数, $\lceil \cdot \rceil$ 表示为取整数.对于高价值和低价值集合的划分也是一样的.

第三步:确定敏捷开发过程中的用例开发顺序.首先开发 U_{hrhv} 集合中的用例,当 $|U_{hrhv}| > 1$ 时,按照用例的风险程度对系统造成的损失估计与该用例的价值之和从高到低进行排序,最先开发值最高的用例.

接下来,将会讨论如何估计UML用例模型中每个用例的价值和风险.

2.1 用例的风险评估

UML序列图是UML模型中用于描述系统动态行为的图形工具之一,它可以对一个用例实例(场景)具体的活动过程进行描述.一个用例可以有多个

场景,所以一个用例可以有多个序列图.通过分析序列图所包含的组件及组件之间的链接情况来度量用例在该场景下的风险程度.在文献[3-5]提出了一种基于UML用例模型的风险评估方法.该评估方法的具体实现流程如图2所示.

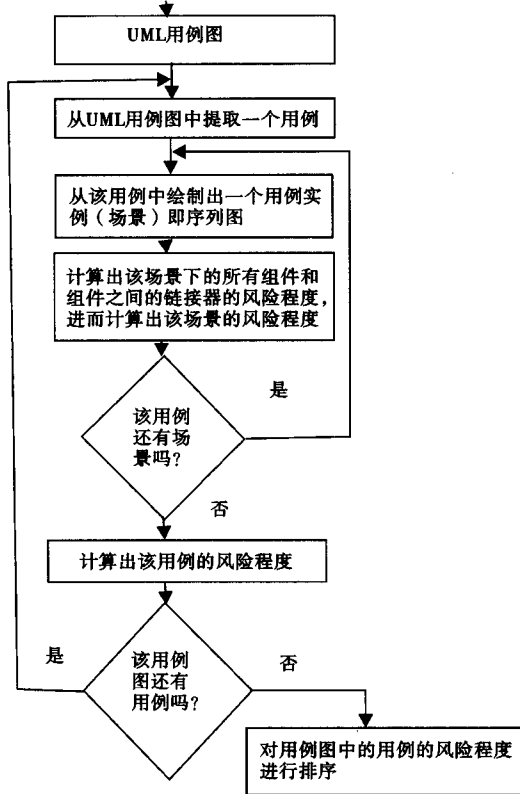


图2 基于UML用例图的风险评估流程

该方法的具体步骤如下:

第一步:度量组件的动态复杂度(即组件失效概率).

$$dco_i^x = t_i^x - c_i^x + 2 \quad (1)$$

式中, dco_i^x 表示组件*i*在某个用例执行场景 S_x 下的动态复杂度; c_i^x 表示的是组件*i*在执行场景 S_x 下被访问过的状态子集的基数; t_i^x 组件*i*在执行场景 S_x 下被访问过的状态子集的基数.对式(1)进行标准化如下:

$$DOC_i^x = \frac{dco_i^x}{\sum_{k \in S_x} dco_k^x} \quad (2)$$

其中 DOC_i^x 是第*i*个组件在执行场景 S_x 下的标准化后的值,它的取值在0到1之间,可以看成是组件*i*的失效概率. $\sum_{k \in S_x} dco_k^x$ 表示在场景 S_x 下所有活动的组件复杂度总和.

第二步:度量组件之间链接器的动态耦合度(即链接器失效概率).

$$EOC_{ij}^x = \frac{|MT_{ij}^x|}{|MT^x|} \quad (i, j \in S_x, i \neq j) \quad (3)$$

EOC_{ij}^x 表示两个组件*i*和*j*之间交互的动态耦合度, MT_{ij}^x 表示场景 S_x 执行期间从组件*i*发送到组件*j*的信息集的基数, MT^x 表示在场景 S_x 执行期间所有活动的组件相互交互的所有信息集的基数,这两个数相除得到的值就是 EOC_{ij}^x , EOC_{ij}^x 的值在0到1之间,它可以用来表示两个组件之间链接器失效的概率.

第三步:确定组件的风险因素.

组件*i*在场景 S_x 中的风险因素 rf_i^x 可以从下式中计算出来:

$$rf_i^x = DOC_i^x * svt_i^x \quad (4)$$

其中 DOC_i^x 是第*i*个组件在场景 S_x 下的标准的复杂度,它的取值 $0 \leq DOC_i^x \leq 1$; svt_i^x 是第*i*个组件在场景 S_x 下的严重等级,它的取值 $0 \leq svt_i^x < 1$;

第四步:确定组件之间链接器的风险因素

用 rf_{ij}^x 表示组件*i*和组件*j*在场景 S_x 下的风险因素分析,可以用下式计算:

$$rf_{ij}^x = EOC_{ij}^x * svt_{ij}^x \quad (5)$$

其中 EOC_{ij}^x 是第*i*个组件与第*j*个组件之间的链接器在场景 S_x 下的标准的耦合度,它的取值 $0 \leq EOC_{ij}^x \leq 1$; svt_{ij}^x 是第*i*个组件和第*j*个组件之间的链接器在场景 S_x 下的严重等级,它的取值 $0 \leq svt_{ij}^x < 1$;

第五步:确定用例场景的风险因素

计算用例场景的风险因素,首先从该用例场景对应的序列图中抽象出控制流图,从控制流图中推导出带吸收状态的离散时间的马尔可夫链,根据马尔可夫链推导出该用例场景下的转移概率矩阵.假设控制图中有一个虚拟的开始状态*S*和五个吸收状态(其中有一个是正常的系统终止状态和四个描述失效状态即分别为次要的、主要的、关键的、灾难性的).如图3表示一个从某个用例执行场景中(即序列图)抽象出来的控制流图.

图中*S*为虚拟的开始状态,*S1*、*S2*、*S3*为瞬态,而*FxMinor*、*FxMajor*、*Fxcriticalr*、*Fxcatastrophic*为用例场景下*x*下的组件的风险严重程度的吸收态,*T*为正常的终止状态.于是离散时间的马尔可夫链对应的概率转移矩阵中的元素计算方法如下:

*S*到*S1*的转移概率为:1

瞬态到正常的终止状态*T*之间的转移概率为:1

瞬态之间的转移概率为:

$$(1 - rf_i^x) * p_{ij}^x * (1 - rf_j^x)$$

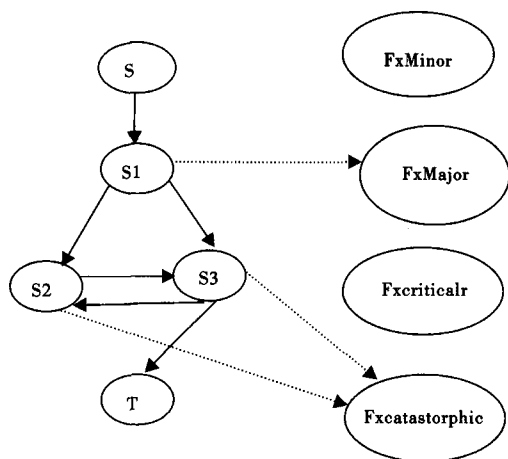


图3 基于用例执行场景的控制流程图

瞬态到风险严重程度的吸收态之间的转移概率

$$\text{为: } rf_i^x + \sum_{k=1}^n (1 - rf_i^x) * p_{ik}^x * rf_k^x$$

其中 rf_i^x 是组件 i 在用例场景 x 条件下失效的概率, p_{ij}^x 表示组件 i 到组件 j 之间的转移概率. 当两个组件成功的转移时, 只有当出发的组件不发生失效即 $(1 - rf_i^x)$ 和两个组件之间的链接器也不发生失效即 $(1 - rf_{ij}^x)$ 的时候才算成功, 只要组件和链接器中有一个发生失效, 则控制流就会转向一个表示风险严重程度的吸收态. 由此可以得出概率转移矩阵. 这个概率转移矩阵是一个 $(n+6)$ 阶矩阵, 其中 n 是表示瞬态数, 有一个虚拟的开始状态数 (如图 4 中 S) 和一个正常终止状态 (如图 4 中 T) 和四个非正常的终止状态数 (如图 4 中的 $FxMinor$ 、 $FxMajor$ 、 $Fxcriticalr$ 、 $Fxcatastrophic$). 对该矩阵从正常终止的状态 T 所在的行列开始把矩阵分成四个部分: Q^x 、 C^x 、 O 、 I . 于是得到一个矩阵方程式:

$$A^x = (I - Q^x) * C^x \quad (6)$$

解上述矩阵方程, 第一行第一列的元素表示系统正常从开始状态 S 到正常终止状态 T 的转移概率, 即在该用例场景下系统成功执行的概率. 所以系统在该用例场景下失效的概率就是用 1 减去上面得到的概率值.

第六步: 确定用例的风险因素.

每个用例 U_k 的风险因素 rf_k 可以通过下式计算出来:

$$rf_k = \sum_{S_x \in U_k} rf^x * P_k^x \quad (7)$$

其中 rf^x 表示用例 U_k 的场景 S_x 的风险因素, P_k^x 表示用例 U_k 的场景 S_x 的出现概率.

2.2 用例的价值评估

在这里考虑的功能的价值就是评估 UML 用例

图中用例的价值. 评估从两个方面考虑: 一个是该用例给系统所带来的正面的好处, 即从该用例的规模、新颖性、创造性、实用性、该用例对整个系统的贡献来评估, 一般是由领域专家、系统分析师、软件开发人员、项目经理、用户等一起研究共同决定. 另一个方面是如果该用例失效会给系统造成的损失程度.

2.3 基于用例的迭代开发顺序的确定

用表格的形式列出每个用例的风险程度和价值, 分别按照用例的风险程度和价值进行排序. 如表 1 所示.

表1 用例的风险程度和价值分析

用例名	风险程度	价值
用例 1	5.6	2000
用例 2	4.2	1500
用例 3	3.5	4000
用例 4	4.5	3000
用例 5	6.2	2800

共有五个用例 $5/2$ 取整后加 1, 可得高风险的用例有 3 个, 而低风险的用例有 2 个, 同样高价值的用例有 3 个, 低价值的用例有 2 个. 如表 2、表 3 所示.

表2 用例的风险程度排序

用例名	风险程度	等级
用例 1	5.6	高
用例 2	4.2	低
用例 3	3.5	低
用例 4	4.5	高
用例 5	6.2	高

表3 用例的价值排序

用例名	价值	等级
用例 1	2000	低
用例 2	1500	低
用例 3	4000	高
用例 4	3000	高
用例 5	2800	高

于是属于高风险高价值的用例有用例 4、用例 5; 属于高风险低价值的用例有用例 1, 属于低风险高价值的有用例用例 3, 属于低风险低价值的有用

例2. 首先开发高风险高价值的用例,但是属于高风险高价值的用例有两个,可以按照这两个用例的价值高低决定先开发用例4,然后依次开发用例5、用例3、用例2、用例1。

3 结束语

本文提出了一种基于UML用例图中各个用例的实际情况来确定迭代策略. 用UML语言建立起来的用例模型能如实地反映出整个系统所要完成的功能. 利用用例的风险程度和价值决定整个敏捷开发过程中的迭代顺序,使高风险高价值的用例优先开发,而对高风险低价值的用例在开发过程中尽量避免,对提高整个系统的可靠性会有很大的帮助。

参考文献:

- [1] Wende Kong. Towards a formal and scalable approach for quantifying software reliability at early development stages[D]. Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, College Park, 2009.

- [2] (美)柯恩,著. 敏捷估计与规划[M]. 宋锐,译. 北京:清华大学出版社,2007.
- [3] Ajith Reddy Guedem. Software architectural risk assessment[D]. Master Thesis submitted to the College of Engineering and Mineral Resources at West Virginia University, 2004.
- [4] Ahmed E Hassan. Architectural level risk assessment [D]. Dissertation submitted to the College of Engineering and Mineral Resources at West Virginia University, 2004.
- [5] Wensheng Hu, Zhouhui, yi Hong. A Method of FTA base on UML use case diagram[C]// ICRMS' [s, l]: IEEE, 2011: 757-759.

作者简介:

胡文生 男, (1969—), 博士生, 讲师. 研究方向为软件可靠性.

赵明 男, (1962—), 博士, 教授, 博士生导师. 研究方向为软件、硬件可靠性、应用统计.

杨剑峰 男, (1986—), 博士生. 研究方向为软件可靠性.

贾国荣 男, (1958—), 教授. 研究方向为电子商务.

(上接第164页)

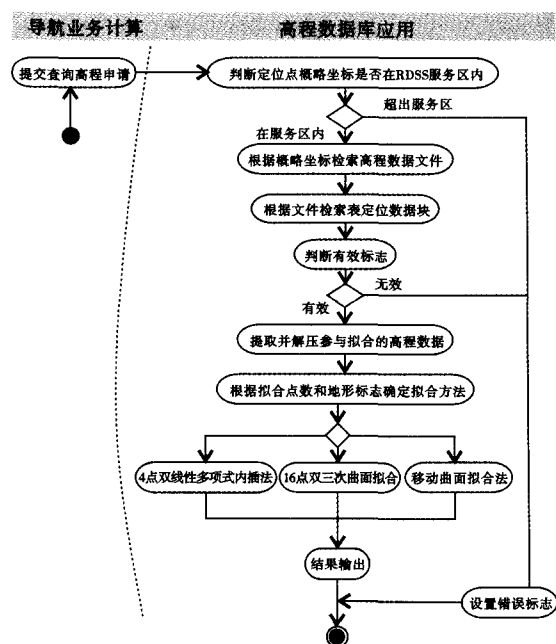


图6 高程数据的提取流程图

从实验结果看出, 本文提出的按地形特征对高程数据分块组织、四叉树索引和文件分级管理模型,

以及双线性内插法、双三次内插法、移动曲面内插法相结合的拟合方法, 可以有效减少高程数据量, 控制精度损失, 并提高数据检索的速度, 满足 RDSS 系统的要求。

参考文献:

- [1] 吴延忠, 李贵琦. 地球同步卫星定位[M]. 北京: 解放军出版社, 1992.
- [2] 谭述森. 卫星导航定位工程[M]. 北京: 国防工业出版社, 2007.
- [3] 陈建鹏. RDSS 高程数据抽取和压缩方法分析[J]. 测绘科学与工, 2009(1): 46-49.
- [4] 张祖勋. 数字摄影测量学[M]. 武汉: 武汉测绘大学出版社, 1996.
- [5] 高扬. 基于 DEM 应用内插算法的分析 [J]. 测绘科技, 1998(2): 23-27

作者简介:

高扬 男, (1969—), 博士研究生, 高级工程师. 研究方向为卫星导航及嵌入式 GIS.

乌萌 女, (1983—), 硕士, 工程师. 研究方向为卫星导航及车辆导航系统.