基于UML模型的敏捷开发迭代顺序的确定

胡文生[1,2,3] 赵 明[3] 杨剑锋[2,3] 龙士工[2]
(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 贵阳 550025)[2] (贵州省可靠性工程中心 贵阳 550025)[3]

摘 要 确定迭代顺序是敏捷开发过程中首先要解决的一个关键问题,它是敏捷开发过程的基础,有很多文献资料在这方面做了大量工作,但都是以功能组的价值为依据来确定迭代开发顺序的,这种以单一指标为依据确定迭代顺序的方式往往会产生一些意料不到的后果,而且对功能组价值的确定大多采用定性的方法,很难用定量的方法来实现。针对敏捷开发过程中迭代顺序研究中所存在的一些问题,提出了利用UML(Unified Modeling Language)用例图和顺序图来计算代表系统各个功能的用例的使用概率和风险程度,以用例的使用概率和风险程度为基础,以概率统计和模糊意见集中决策方法为手段来定量确定敏捷开发过程中的迭代顺序。

关键词 UML,用例,风险程度,价值,迭代顺序
中图法分类号 TP312 **文献标识码** A

Determination of Agile Development Iteration Order Based on UML

HU Wen-sheng[1,2,3] ZHAO Ming[3] YANG Jian-feng[2,3] LONG Shi-gong[2]
(Department 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 To determine the iteration order is a critical problem in the agile development, and it is the foundation of agile development process. There are a lot of literatures to do a lot of work in this area. But the iteration order is based on the value of functional groups in many literatures. This iteration order based on a single indicator will often be unexpected consequences. The value of the functional groups is mostly determined by the qualitative approach, and it is difficult to using the quantitative methods to realize it. This paper put forward that we can use UML(Unified Modeling Language) use case diagram and sequence diagram to calculate the use probability and risk degree of use cases representing each system functional, to solve the problems about the study of iterative sequence in the agile development process. And based on the use probability and risk degree of cases, we can determine iterative sequence in the agile development process with the method of Probability and Statistics and Fuzzy Decision-making Method for opinion concentrated.

Keywords UML, Use case, Risk, Value, Iteration order

1 引言

随着信息技术的广泛应用,软件规模变得越来越大,用户的需求变化极快,有可能软件产品还没有开发出来,用户的需求已经彻底地发生了改变,从而导致软件项目的失败。这种失败的原因既有用户在软件开发初期需求不确定的因素,也有开发组织没有及时与用户沟通的因素。所以对于一个软件组织来说,要想保证软件项目开发的成功,及时与用户沟通,了解用户的需求变化,根据需求变化适时做出调整,积极响应用户的需求变化是很有必要的,毕竟满足用户的需求是软件开发的最终目标。而传统的软件开发方法恰恰是限制系统开发人员与用户之间的沟通,用户只是在系统开发的前期即软件需求分析阶段和系统开发完成之后的确认测试和验收测试阶段才参与进来,而在项目开发过程中,即软件设计、实现、测试阶段前期(比如单元测试、集成测试、系统测试),用户完全被排除在外。这种传统的软件开发方式存在一个致命的缺陷就是一旦软件产品开发完成之后,如果在确认测试和验收测试阶段才发现软件产品与用户所期待的产品之间存在很大的差距,这个时候去修改软件,无论是开发成本,还是公司的信誉,开发方都要承担巨大的损失。为了避免这种情况的发生,现在流行的软件开发思想是让用户全程参与到软件开发的每一个阶段,这种思想就是敏捷开发。在20世纪90年代末,软件开发方法领域的一些专家提出了如极限编程、自适应软件开发、动态系统开发等一系列敏捷开发方法。这些敏捷开发

到稿日期:2013-03-21 返修日期:2013-06-14 本文受国家自然科学基金(61163001)资助。
胡文生(1969-),男,博士生,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和硬件可靠性、应用统计;杨剑锋(1986-),男,博士生,主要研究方向为软件可靠性;龙士工(1965-),男,博士,教授,硕士生导师,主要研究方向为信息安全。

方法都有一个共同的特点，就是重视用户的需求变化。但是如果敏捷开发仅仅是把积极响应用户的需求放在软件开发的第一位，这样开发出来的软件产品虽然能很好地满足用户的需求，但未必是用户最满意的产品，也就是说，如果软件产品的质量得不到保证，这样的软件产品也很难让用户接受。为了既能满足用户的需求目标，又能满足用户的质量目标，在敏捷开发过程有必要考虑系统的安全性、可靠性、可维护性等质量属性，综合多种因素而不是单一因素来决定敏捷开发的迭代顺序。

本文第1节主要介绍一些背景知识；第2节介绍敏捷开发过程中的迭代顺序；最后对全文进行总结并对未来的研究方向进行探讨。

2 敏捷开发过程中迭代顺序的规划

敏捷开发方法采用循环、迭代的增量式开发方式，把用户的需求逐次添加到软件产品中，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，获得用户的反馈意见，及时发现与需求不一致的地方。下一次迭代开发是以用户新的需求、上次迭代产生的软件产品和上一次的反馈意见为基础，这样不断循环下去，最后交付给用户的必定是最接近用户期待的软件产品。在整个迭代过程中，需要解决一个关键问题，就是哪些用户的需求先开发，哪些用户的需求后开发，即如何确定敏捷开发的迭代顺序。大部分文献资料都是把用户的需求表示成系统的功能，再以功能的重要程度来决定敏捷开发的迭代顺序。在文献[1]中阐述了将系统核心功能放在敏捷开发的前期能显著提高软件系统的可靠性，并给出了相应的计算公式：

$$\hat{R}_k^{(i)}=1-(1-P)^{k-i}*(1-w\hat{R}_1^{(i)}) \tag{1}$$

式中，$\hat{R}_k^{(i)}$ 表示第 i 个功能组在第 k 次迭代所产生的产品的可靠性估计值，P 表示纠错率，$w(w\in[0,1])$表示测试的充分性。然后再通过式(2)可以计算出敏捷开发最终软件产品的可靠性[1]：

$$\hat{R}_s=1-\sum_{i=1}^{k}C_i*(1-\hat{R}_k^{(i)}) \tag{2}$$

式中，$\hat{R}_s$ 是最终软件产品的可靠性估计值；$C_i=\dfrac{N_i}{\sum_{i=1}^{k}N_i}(i=1,2,\cdots,k)$，其中 N_i 表示系统不同功能组初始测试时的测试用例数，C_i 表示每次迭代产品的可靠性权重。

利用上述式(1)、式(2)计算某个具有6个不同优先级别功能组的敏捷开发项目的可靠性。

表1 各个用例权重的确定

功能组	N_i	S_i	R_i	$w*R_i$	C_i
F1	210	178	0.848	0.7632	0.1941
F2	192	166	0.865	0.7785	0.1774
F3	185	150	0.811	0.7299	0.1710
F4	183	152	0.831	0.7479	0.1691
F5	172	145	0.843	0.7587	0.1590
F6	140	121	0.864	0.7776	0.1294

注：数据来源于文献[1]。

表1中功能组是经过优先级别排过序的，即功能组F1表示整个项目中最核心的功能组，是最先进行迭代的功能组，总共需要6次迭代，N_i 表示测试用例数，S_i 表示测试成功数，R_i 表示功能组的可靠性点估计。于是可以计算出各个功能组经过6次迭代后的可靠性变化情况，如表2所列。

表2 各功能组在每次迭代周期的可靠性

次数	F1	F2	F3	F4	F5	F6
1	0.7632					
2	0.8698	0.7785				
3	0.9284	0.8782	0.7299			
4	0.9606	0.9330	0.8514	0.7479		
5	0.9783	0.9631	0.9183	0.8613	0.7587	
6	0.9881	0.9797	0.9551	0.9237	0.8673	0.7776
R_s						0.9236

注：数据来源于文献[1]。

从上述敏捷项目的可靠性计算过程中可以看出，将项目的核心功能组放在迭代周期的前面，经过多次迭代之后可以显著提高核心功能组的可靠性，从而提高整个项目的可靠性。由于迭代的顺序是由各个功能组的价值来确定的，因此各个功能组的价值确定方法就显得尤为重要。但是在文献[1]中并没有给出核心功能组的确定方法，而且如果仅以各个功能组的价值来确定敏捷开发的迭代顺序，有可能会出现这样的情况：当软件开发小组开发程序大部分功能时，在开发后期某次迭代过程中突然遇到一个有很大风险的功能组，从而有可能危及到整个软件产品的交付时间和软件产品的质量，而且在迭代最后几个阶段测试和修改都无法得到充分保证，匆忙交付给用户，可能会造成用户对产品的满意度下降。所以仅仅考虑软件功能组的价值而不考虑功能组的风险来确定迭代顺序，其效果并不理想。为了解决这种情况，在文献[2,6]中提出按照风险-价值关系来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4个象限中。

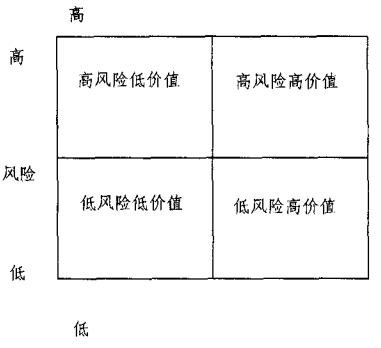

图1 软件系统功能风险价值图

根据图1确定迭代顺序为：高风险高价值的功能组→低风险高价值的功能组→低风险低价值的功能组。尽可能把高风险、低价值的功能组推迟到项目之外，没必要为低价值的功能组冒很大的风险。把高风险高价值的功能组放在敏捷开发的第一个迭代周期去实现，目的是在随后多个迭代周期中不断发现它所存在的问题，不断纠正各种缺陷。这样做的好处是不会造成对各种不同的功能组平均分配开发资源，而是把更多的资源应用到那些具有高风险高价值的功能组上，从而既提高了整个系统的可靠性，又优先实现了系统的核心功能，让用户最关心的核心功能最先实现。

上述观点比文献[1]有了一定程度的进步，在确定敏捷开发的迭代周期过程中不仅考虑了各个功能组的价值，而且考虑了各个功能组的风险程度。但是文献[2]依然没有给出确定各个功能组价值和风险的具体方法。如果仅仅是通过与客户沟通来确定各个功能组的价值和风险，主观随意性比较强，

而且用户所关心的核心功能组可能会有很多，各个功能组的风险程度事先用户也并不是很清楚。所以需要一种能定量确定各个功能组的价值和风险的新方法和技术。在研究 UML 各个模型图过程中发现用例图是对系统所包含功能情况的一种很好的表述方式，用例图一般是由用例、参与者、系统边界组成，其中每一个用例就是对系统某一功能的通用描述。每个用例的实现是通过对象之间一系列相互交互来完成的，而描述这种交互情况的 UML 模型图是顺序图。所以对系统功能组的价值和风险评估，完全可以通过 UML 用例图和顺序图来实现。接下来，将会讨论如何通过 UML 用例图和顺序图来评估每个用例的价值和风险。

2.1 基于 UML 用例图的用例价值的评估

用例图描述一个系统所能提供的全部功能情况，反映了外部参与者如何与系统进行交互以及用例与用例之间的相互关系，图 2 是一个简单的用例图。

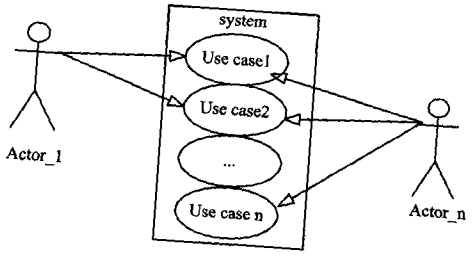

图 2 UML 用例图

在用例图中可以为每个参与者标上它们的出现概率，标注每个参与者使用各个用例的概率。图 3 描述了一个简单的加了标注的用例图，在该用例图中有 m 种类型的用户、n 个用例，加标注的方式类似于文献[3,5]。

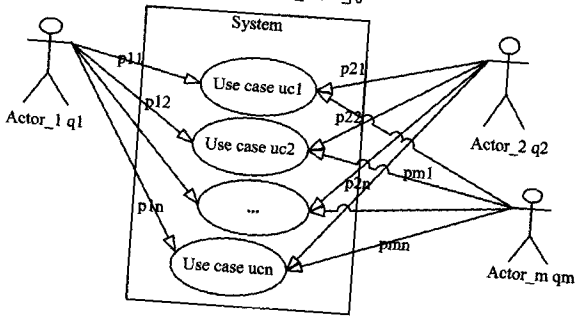

图 3 加标注的 UML 用例图

在图 3 中，$q_1, q_2, \cdots, q_m$ 表示参与者 $Actor\_1, Actor\_2, \cdots, Actor\_m$ 出现的概率且 $\sum_{i=1}^{m} q_i = 1$，p_{ij} 表示第 i 类参与者 $Actor\_i$ 使用 UML 用例图中的第 j 个用例的概率且 $\sum_{j=1}^{n} q_{ij} = 1$。这是在用例图中某个用例 j 被所有参与者使用的概率计算公式为：

$$p(j) = \sum_{i=1}^{m} p_i * p_{ij} \tag{3}$$

利用式(3)将 UML 用例图中的所有用例的使用概率全部计算出来，把用例的使用概率看成是该用例所代表的功能的价值，概率值高的则价值高，概率值低的则价值低。

基于 UML 顺序图的用例风险的评估

UML 用例图包含的用例表示系统的功能，而功能的实现需要一系列对象或组件相互交互、相互通信来具体完成。这种相互交互、相互通信的图就是 UML 序列图(见图 4)，也叫顺序图，一个序列图可以看成是一个用例的执行场景，有些情况下一个用例可以有多个执行场景。通过分析序列图中所包含的对象或组件及它们之间的交互情况来度量用例在该场景下的风险程度。文献[3,4]提出了一种基于 UML 序列图的风险评估方法。

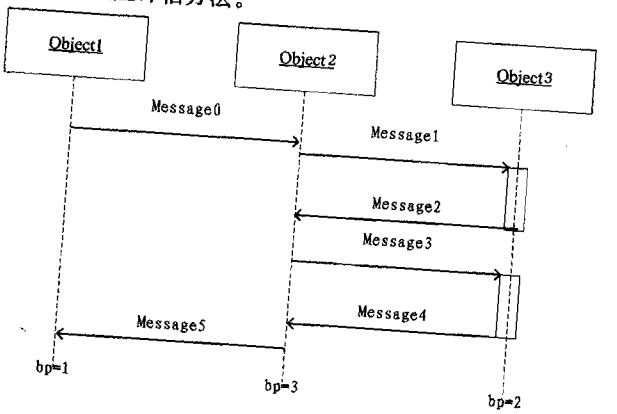

图 4 UML 序列图

把一个交互消息进入一个对象生命线和交互消息离开对象生命线之间的时间间隔称为该对象的繁忙期。从图 4 中可以看出对象 *Object2* 繁忙期有 3 个，繁忙期用 bp 表示，则对象 *Object2* 对应的 $bp=3$。对象只有在使用过程中才会发生失效或故障，用 $bp\ ij$ 表示对象 *Objecti* 在用例对应的序列图 j 中的繁忙期的数目，θi 表示对象 *Objecti* 的失效概率，θij 表示对象 *Objecti* 在序列图 j 的失效概率，则 θ_{ij} 的计算公式如下[3,4]：

$$\theta_{ij} = 1-(1-\theta_i)^{bp_{ij}} \tag{4}$$

于是可以通过式(4)计算出各个用例的失效概率。假设某个用例 x 有 k 个序列图，第 k 个序列图中包含有 m_k 个对象相互交互，则用例 x 的失效概率的计算公式为：

$$p(x) = \sum_{j=1}^{k} \sum_{i=1}^{m_k} (1-(1-\theta_i)^{bp_{ij}}) \tag{5}$$

根据式(5)可以计算出用例的失效概率，把该用例的失效概率看成是该用例的风险程度。

2.3 基于用例的价值和风险程度确定迭代顺序

通过上面的有关知识可以计算出系统中所有用例的使用概率和失效概率。把用例的使用概率看成是用例的价值，把用例的失效概率看成是用例的风险程度。利用模糊意见集中决策的方法对已经确定价值和风险程度的用例进行排序，从而确定敏捷开发过程中的迭代顺序。模糊意见集中决策的基本原理如下[7]：假如对论域 $U=\{u_1, u_2, \cdots, u_n\}$ 中的元素进行排序，先由 m 个专家组成专家小组 M，分别对 U 中的元素进行排序，不同的专家按照自己的标准进行排序，从而得到 m 种意见 $V=\{v_1, v_2, \cdots, v_m\}$，其中 v_i 是对论域 U 中的 n 个元素所做的第 i 种排序方案，若 u_j(其中 $j=1,2,\cdots,n$)在第 i 种排序方案 v_i(其中 $i=1,2,\cdots,m$)中排在第 k 位，设 $B_i(u_j)=n-k$，则将式(6)计算出来的 $B(u_j)$ 称为 u_j 的 Borda 数。论域 U 中的所有元素都可以按照 Borda 数的大小进行排序。

$$B(u_j) = \sum_{i=1}^{m} B_i(u_j) \tag{6}$$

如果对所有的排序方案中的某个排序方案更感兴趣，可以在 $B_i(u_j)$ 前面乘上一个较大的权重系统 a_i(其中 $\sum_{i=1}^{m} a_i = 1$)。于是式(6)变成如下的一个公式：

$$B(u_j) = \sum_{i=1}^{m} a_i * B_i(u_j) \tag{7}$$

假如有一个系统 UML 用例图中所包含的用例风险程度和价值的情况如表 3 所列。

表 3　某个 UML 用例图中各个用例的风险程度和价值

用例名	风险程度	价值
uc1	7.6	0.78
uc2	4.2	0.92
uc3	3.5	0.45
uc4	4.5	0.42
uc5	5.2	0.62
uc6	6.2	0.82

表 3 包含 6 个用例，按照价值可以将整个系统的用例排序为：*uc*2、*uc*6、*uc*1、*uc*5、*uc*3、*uc*4；按照用例的风险程度可以将整个系统的用例排序为：*uc*1、*uc*6、*uc*5、*uc*4、*uc*2、*uc*3。根据式(6)有：

$B(uc1)=(6-3)+(6-1)=8$

$B(uc2)=(6-1)+(6-5)=6$

$B(uc3)=(6-5)+(6-6)=1$

$B(uc4)=(6-6)+(6-4)=2$

$B(uc5)=(6-4)+(6-4)=4$

$B(uc6)=(6-2)+(6-2)=8$

所以得出的敏捷迭代顺序为：*uc*1、*uc*6、*uc*2、*uc*5、*uc*4、*uc*3。

如果开发人员和用户更看重用例的价值，可以利用式(7)，对用例的价值对应的权重取更大一些，比如取 0.8，则风险程度的权重就取 0.2，于是：

$B(uc1)=0.8*(6-3)+0.2*(6-1)=2.8$

$B(uc2)=0.8*(6-1)+0.2*(6-5)=4.2$

$B(uc3)=0.8*(6-5)+0.2*(6-6)=0.8$

$B(uc4)=0.8*(6-6)+0.2*(6-4)=0.4$

$B(uc5)=0.8*(6-4)+0.2*(6-4)=2$

$B(uc6)=0.8*(6-2)+0.2*(6-2)=4$

所以得出的敏捷迭代顺序为：*uc*2、*uc*6、*uc*1、*uc*5、*uc*3、*uc*4。

如果开发过程中更看重用例的风险程度对产品和用户的影响，则可以设置风险程度的权值更大一些，其计算过程与上面类似。

结束语　本文通过研究得出如下的结论：

• 将敏捷开发迭代顺序从功能组为基础转向以 UML 用例图中的用例为基础。

• 改变了以往敏捷开发迭代顺序以单一指标为依据的缺陷，本文利用用例的风险程度和使用概率来综合考虑敏捷开发的迭代顺序。

• 以往敏捷开发迭代顺序的确定大多是以定性方法为主，主观随意性比较强。本文利用概率统计方法和模糊意见集中决策方法来定量确定敏捷开发的迭代顺序。

总之，本文提出的方法能够为软件开发人员在确定迭代顺序时提供一种定量的决策依据，而且可以根据用户和开发人员的关注点来调整项目的开发顺序，如果项目更重视可靠性，则风险因素成为重要考量的因素，如果项目更重视用户的使用情况，则价值成为主要关注的因素，即该方法能很好地反映出项目开发的迭代顺序随项目涉众人员关注点的不同而不同，使用起来比较灵活。

参 考 文 献

[1] 王晓华. 敏捷开发环境下软件可靠性分析及相关问题研究[D]. 贵阳：贵州大学，2008：49-54

[2] (美)柯恩. 敏捷估计与规划[M]. 宋锐，译. 北京：清华大学出版社，2007：95-200

[3] Cortellessa V. Harshinder Singh and Bojan Cukic. Early reliability assessment of UML based software models[C]// WOSP '02. Rome, Italy, July 2002：24-26

[4] 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

[5] 胡文生，赵明，杨剑锋. 一种基于 UML 用例模型的软件可靠性分配方法[J]. 计算机科学，2012，39(6A)

[6] 胡文生，赵明，杨剑锋，等. 敏捷开发过程中的迭代策略分析[J]. 微电子学与计算机，2012，29(5)

[7] 侯福均，吴祈宗. 模糊偏好关系与决策[M]. 北京：北京理工大学出版社，2009(2)：72-77

[8] Johnstone C P, Lill A, Reina R D. Does habitat fragmentation cause stress in the agile antechinus? A haematological approach [J]. Journal of Comparative Physiology B: Biochemical, Systemic, and Environmental Physiology, 2011, 182(1): 139-155

[9] Mikulenas G, Kapocius K. An Approach for Prioritizing Agile Practices for Adaptation [M]. Information Systems Development, 2011, Part 7: 485-498

[10] 江瑜. 基于 UML 的敏捷建模方法研究[J]. 计算机工程与设计，2008，29(15)

[11] 段隆振，王凤斌，甘晟科，等. 基于敏捷化统一过程需求建模的研究及实践[J]. 计算机科学，2006，33(10)

[12] Limpens F, Team R, EPFL, et al. Towards agile competence development[C]// 2011 IEEE International Conference on Date of Conference. March 2012：667-672

[13] Swaminathan B, Jain K. Implementing the Lean concepts of Continuous Improvement and Flow on an Agile Software Development Project: An Industrial case Study[C]// AGILE India (AGILE INDIA). Feb. 2012：10-19

[14] Bustard, David. Beyond Mainstream Adoption: From Agile Software Development to Agile Organizational Change[C]// Engineering of Computer Based Systems (ECBS), 2012 IEEE 19th International Conference. April 2012：90-97

(上接第 196 页)

[17] Quinlan J R. Bagging, boosting and C4. 5[C]// Proceedings of the National Conference on Artificial Intelligence. 1996：725-730

[18] Siraj M M, Maarof M A, Hashim S Z M. A Hybrid Intelligent Approach for Automated Alert Clustering and Filtering in Intrusion Alert Analysis[J]. Journal of Computer Theory and Engineering, 2009, 1(5)：539-45

[19] Panda M, Abraham A, Patra M R. A Hybrid Intelligent Approach for Network Intrusion Detection[J]. Procedia Engineering, 2012, 30：1-9

[20] http://www.ll.mit.edu/mission/communications/ist/corpara/ideval/data

[21] http://tools.netsa.cert.org/SiLK

[22] Witten I H, Frank E. Data Mining: Practical Machine Learning Tools and Techniques (second ed.) [M]. Morgan Kaufmann Publishers, 2005