ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.10, October 2007, pp.2380−2392
DOI: 10.1360/jos182380

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一个三维需求模型及其对涉众协同的支持*

王继喆[1,2], 李明树[1,3+]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 研究生院,北京 100049)

[3](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100080)

A Tri-Dimensional Requirements Model and Its Support for Stakeholder Coordination

WANG Ji-Zhe[1,2], LI Ming-Shu[1,3+]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62661001, E-mail: mingshu@iscas.ac.cn, http://www.iscas.ac.cn

Wang JZ, Li MS. A tri-dimensional requirements model and its support for stakeholder coordination. *Journal of Software*, 2007,18(10):2380−2392. http://www.jos.org.cn/1000-9825/18/2380.htm

Abstract: In collaborative software development, WinWin state is hard to achieve because stakeholders are concerned about different aspects of software development such as technology, human, and process. Frequently the influences among the win conditions of stakeholders are implicit thus the conflicts are hard to find. This paper uses a tri-dimensional requirements model, named TRISO-RM, to describe the win conditions of stakeholders on different aspects of software development. TRISO-Elements, each of which is formed by interconnected actors, an artifact, and an activity, are used as the medium to find, build and maintain the relationships among stakeholder win conditions. The production mechanism of model clashes is discussed and the process to find and avoid them is presented based on TRISO-RM. In particular, the application in the development of SoftPM is described to illustrate and validate the TRISO-RM method.

Key words: requirements model; tri-dimensional requirements model; stakeholder coordination; model clash; win condition

摘 要: 在多涉众(stakeholder)参加的协同开发活动中,不同的涉众关注软件开发的不同方面,例如技术、过程、人.由于涉众的“赢条件(win condition)”之间的相互影响冲突往往是隐含的,所以,它们之间的冲突不容易被发现,导致各方共赢的均衡状态难以达成.提出了一个三维的需求模型 TRISO-RM(tri-dimensional integrated software requirements model),通过它来描述和集成涉众对于软件开发不同方面的赢条件.在此基础上,通过由一组互相依赖的制品、活动、参与者所构成的 TRISO-Element 作为媒介,以发现、建立和维护不同赢条件之间的关系.基于 TRISO-RM,分析了涉众目标冲突的一种典型形式——模型冲突(model clash)所发生的机理,并给出了发现与解

* Supported by the National Natural Science Foundation of China under Grant Nos.60273026, 60573082 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA113080 (国家高技术研究发展计划(863))

Received 2006-03-06; Accepted 2006-06-12

决的过程.通过在软件质量管理平台 SoftPM 的开发中的实践应用,TRISO-RM 方法对于涉众协同支持的有效性得到了验证.

关键词: 需求模型;三维需求模型;涉众协同;模型冲突;赢条件

中图法分类号: TP311 **文献标识码:** A

软件开发是一个多涉众(stakeholder,也有译作“利益相关者”)参与的过程,不同的涉众在这一过程中的目标各不相同.在Theory W[1]中,Boehm将这些目标称为“赢条件(win condition)”,因为涉众将它们的实现与否作为判断自己是否“赢”的依据.Boehm还指出,软件开发成功的充分必要条件是所有的涉众的“赢条件”都得到满足,也即到达共赢(WinWin)状态[1].

赢条件之间不产生冲突是共赢的基本前提.在过去十几年中,研究者们的工作主要着眼于用户(user)的赢条件,即软件需求的冲突检查和处理上,较有代表性的如基于逻辑的检测[2]、基于模型检验的检测[3]、多视角需求工程方法[4,5]、基于目标的方法[6−8]以及新近提出的面向方面的需求工程方法[9,10].文献[11,12]对这些方法进行了综述.另一类典型的研究关注于冲突发现和处理的整体框架或者过程,例如Boehm等人提出的EasyWinWin方法[13,14]、Nuseibeh等人提出的需求不一致性的管理模式[15]、Robinson等人提出的协商生命周期[16]等.在这些方法中,对需求冲突的发现都需要满足一个前提,即冲突的需求所关注的应该是同一个领域中的相同的事物[12,17],在文献[11]中将其称为需求“指称域的重叠”.在实践中人们发现:由于用语的混乱以及缺乏一个一致的语义框架,这种重叠关系常常会被遗漏[13].为了解决这个问题,在Oz系统[18]和EasyWinWin中,采用了统一的领域模型来建立参与者对术语及其关系的一致理解和使用.

当考察的内容扩大到多种涉众的赢条件时,其情景更为复杂:首先,由于各涉众角色、目标不同,他们的赢条件之间的冲突所涉及的范围更加广泛;其次,当其赢条件所描述的是软件开发的过程、技术等不同方面时,其冲突往往是隐含的,难以直接发现.例如:当过程管理者希望采用基于瀑布模型的软件过程(赢条件 *A*),而开发人员希望使用 COTS(commercial-off-the-shelf,商业构件)技术(赢条件 *B*)时,COTS 技术的使用,决定了需要使用特殊的(非瀑布模型的)软件过程(隐含的赢条件 *C*),这与赢条件 *A* 是冲突的.这类冲突在根本上都是由于涉众们看待事物的“心智模型”即认知方式和角度的不同而造成,所以在文献[19,20]中,将其称为模型冲突(model clash).

消除模型冲突是实现涉众共赢的必要条件,从而也是软件项目成功的基本要素.一种方法若要有效地支持模型冲突的发现和消除,必须能够对赢条件冲突的产生机理给出清晰的解释.由于赢条件所描述的是对软件开发不同方面的各类要素的期望,因此,该方法应该能够揭示这些要素之间相互影响的机制.在本文所描述的方法中,我们通过一个三维的需求模型来描述涉众在不同方面上的赢条件,通过三维元素之间的交互关系来建立和检查涉众的赢条件之间的关系,并进一步给出模型冲突发现和解决的机制.我们在软件质量管理平台SoftPM[21]的开发过程中对其进行了实践应用,验证了其对涉众协同的支持的有效性.

1 TRISO-RM

在文献[22]中,我们通过 TRISO-Model(tri-dimensional integrated software development model,三维软件开发模型)将软件开发中涉众所关注的要素划分到技术、过程、人三个维度上.由于赢条件是涉众对软件开发中某些要素的特定属性的期望,因此,根据其关注的要素及其属性,赢条件也可以划分到这 3 个维度.在本文中,针对涉众协同的问题,我们进一步提出一个三维需求模型 TRISO-RM(tri-dimensional integrated software requirements model),并对 3 个维度所涵盖要素进行描述,以此作为对赢条件之间关系进行分析,从而发现和消除模型冲突的基础.图 1 是对 TRISO-RM 所描述的软件开发的 3 个维度的示意性的描述.对于图 1 中的被观察的点来说,观察点表现的是一个集成的视图,由 3 个相关维度上的属性所共同决定.当涉众分别从技术、过程、人三个单一维度对软件开发进行观察的时候,所获得的视图是 3 个不同的一维视图,分别包含了某一特定方面的不完整的信息.

具体地,在技术维上,涉众所关注的是 Artifact(制品)的开发技术、工具、Artifact 之间的关系等,即以 Artifact

为中心的视图.在过程维上,涉众关心的是以 Activity(活动)为基本单元的软件过程的组织、管理、度量等因素.在人维上,涉众关心的是和 Actor(参与者)相关的能力、组织、经济方面的因素.

涉众的视图往往是由这些视图的某些部分组合而成的,它们在描述的内容上可能并无重合之处.但是,由于它们所反映的是同一个(组)观察点在不同维度上的表现,它们之间的交互关系可以通过集成视图发现.在第 1.2 节中我们将进一步讨论.

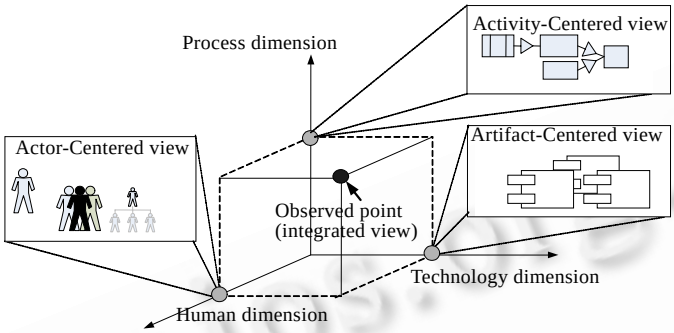

Fig.1 Three dimensions of software development in TRISO-RM

图 1 TRISO-RM 中软件开发的三维

1.1 TRISO-RM的描述

定义 1(制品Artifact). 制品是软件开发过程中制造的文档、模型或其他有形实体,*Artifact*=(*arfType*,*arfSize*,*arfTechniques*,*arfStatus*),其中:*arfType*是制品的类型;*arfSize*是制品规模,其单位视制品类型而定;*arfTechniques*={$arfTechnique_i|i=1...n$}是构建该制品所需要的技术集合;*arfStatus*是制品的状态.制品的全体记作*Artifacts*.

定义 2(活动Activity). 活动是软件开发中的最小有形工作单元,具有明确定义的输入和输出,*Activity*={*atvType*,*atvStatus*,*atvDuration*,*atvInput*,*atvOutput*},其中:*atvType*是对活动的分类;*atvDuration*是活动持续时间;*atvStatus*说明活动的状态;*atvInput*={$a_i|a_i\in Artifacts,i=0...n$}和*atvOutput*={$a_j|a_j\in Artifacts,j=1...n$}分别为输入和输出制品的集合.活动的全体记作*Activities*.

定义 3(参与者Actor). 参与者是执行软件开发过程的人,*Actor*=(*atrRole*,*atrCapabilities*,*atrEffort*),其中:*atrRole*是参与者的角色;*atrCapabilities*={$atrCapability_i|i=1...n$}是参与者的能力集合;*atrEffort*是参与者的工作量.参与者的全体记作*Actors*.

定义 4(三维需求模型 TRISO-RM). TRISO-RM 是一个 5 元组(*EReqs*,*T-Features*,*P-Features*,*H-Features*,*Interactions*),其中:

- *EReqs*=(*Reqs*,*ERQuality*,*ERSize*)是原始需求,即对软件系统的期望特性的描述.其中:原始需求项集合*Reqs*={$requirement_1,\ldots,requirement_n$};*ERQuality*是对原始需求的质量属性描述集合*ERQuality*=(*clarity*,*completeness*,*changeability*);规模和复杂度度量集合*ERSize*=(*physicalsize*,*complexity*);
- *T-Features*=(*Artifacts*,*ArfsAttributes*,*R-Arfs*,*Ext-T-Elements*)是技术维属性集合,其中:*Artifacts*见定义 1;*ArfsAttributes*={$ArfAttribute_i|i=1...n$}是*Artifacts*的属性集合;*R-Arfs*定义*Artifact*之间的二元关系*R-Arfs*:*Artifacts*×*Artifacts*→*R-Arfs-Types*;*Ext-T-Elements*是扩展的技术维元素*Ext-T-Element*的集合,*Ext-T-Element*=(*eteDescription*,*eteR2arfs*),定义这类元素及其与制品(或其关系)的关系;
- *P-Features*=(*Activities*,*AtvAttributes*,*R-Atvs*,*Ext-P-Elements*)是过程维属性集合,其中:*Activities*见定义 2;*AtvAttributes*是*Activities*的属性集合,*AtvAttributes*={$AtvAttribute_i|i=1...n$};*R-Atvs*定义活动之间的二元关系*R-Atvs*:*Activities*×*Activities*→*R-Atvs-Types*;*Ext-P-Elements*是扩展的过程维元素*Ext-P-Element*的集合,*Ext-P-Element*=(*epeDescription*,*epeR2atvs*),定义这类元素及其与活动(或其关系)的关系;
- *H-Features*=(*Actors*,*AtrAttributes*,*R-Atrs*,*Ext-H-Elements*)是人维元素集合,其中:*Actors*见定义 3;

*AtrAttributes*是*Actors*的属性集合,*AtrAttributes*={*AtrAttribute*$_i$|*i*=1...*n*};*R-Atrs*定义*Actor*之间的二元关系*R-Atrs*:*Actors*×*Actors*→*R-Atrs-Types*;*Ext-H-Elements*是扩展的人维元素*Ext-H-Element*的集合,*Ext-H-Element*=(*eheDescription*,*eheR2atrs*),定义这些元素及其与参与者(或其关系)的关系;

- *Interactions*=(*R-Inter-Features*,*R-EReqs-Features*)是维间关系集合,其中:*R-Inter-Features* 定义技术、过程、人三个维度的元素之间的关系;*R-EReqs-Features* 定义各维度元素与原始需求之间的关系.

1.2 各维之间的相关性

TRISO-RM 的三维之间既相对独立又具有密切的联系.涉众在从不同视角对软件开发进行观察时,往往对TRISO-RM 的过程、技术、人三个维度分别独立对待.但是,3 个维度在本质上又是不可分割的,必须协调存在,否则软件开发将不能有效地进行.这种联系的存在,是因为三维的核心 *Actor*,*Artifact*,*Activity* 是互相依存的:

1) 在软件开发中,任何一方都不能脱离另外二者存在,否则:① 要么,这种存在没有意义,例如:如果 *Actor* 在开发中不执行任何 *Activity*,则可以将其排除在外;② 要么不可能实现,例如:没有 *Actor*,*Activity* 在过程中无法执行.

2) 任何一方所表现出来的属性都受到其他二者属性的影响.例如:当一个 *Actor* 执行一个 *Activity* 的时候,*Actor* 的效率、能力都会对这个 *Activity* 的执行效果产生影响.

由于这种依存性,*Artifact*,*Activity* 和 *Actor* 作为一个整体存在,我们对其定义如下:

定义 5(三维基本元素 TRISO-Element). *TRISO-Element*={*arf*,*atv*,*atrs*},其中:*arf*∈*Artifacts*;*atv*∈*Activities*;*atrs*⊆*Actors*;*arf* 是 *atv* 的输出制品;*atrs* 是 *atv* 的执行者集合.三维基本元素的全集记作 *TRISO-Elements*.

下面,我们以定理的形式给出 *Artifact*,*Activity* 和 *Actor* 之间主要的相关性(relevance).这些相关性的成立是软件开发顺利进行的基础,即在一个顺利进行的软件开发中,下面这些定理成立.

定理 1(Artifact-Activity-Actor 存在相关性). 对于同一个项目中的 *Activities*,*Artifacts*,*Actors* 集合,有

$\forall a \in Activities, \exists a' \in Artifacts(a' = a.output); \forall a' \in Artifacts, \exists a \in Activities(a' = a.output)$

$\forall a \in Activities, \exists a'' \in Actors(a'' \in a.executer); \forall a'' \in Actors, \exists a \in Activities(a'' \in a.executer)$

$\forall a' \in Artifacts, \exists a'' \in Actors(a'' \in a'.auther); \forall a'' \in Actors, \exists a' \in Artifacts(a'' \in a'.auther)$

证明:存在相关性即是说明 *Actor*,*Artifact*,*Activity* 任何一方都不能脱离另外两者而存在,否则不能存在或者没有存在的意义,可以从项目中删除. □

定理 2(Artifact-Actor 资源相关性). 在一个 TRISO-Element 中,*arf* 规模的增减将引起 *atrs* 工作量的增减.

证明:根据COCOMOII[23]中给出的工作量估算公式 $Effort = A \times Size^E \times \prod_{i=1}^{n} EM_i$,其中:*A*,*E*, $\prod_{i=1}^{n} EM_i$ 在这里可以看作大于 0 的常量.因此,*atrs* 的工作量是 *arf* 的规模的单调递增函数,定理得证. □

定理 3(Artifact-Activity 时间相关性). 在一个 TRISO-Element 中,*atrs* 不变的情况下,*arf* 规模的增加将引起 *atv* 的持续时间延长.

证明:根据COCOMOII[23]中给出的开发周期估算公式*Duration*=*C*×(*Effort*)^F,其中:*C*,*F*是大于 0 的常量.根据定理 2,*atv*的持续时间是*arf*的规模的单调递增函数,定理得证. □

定理 4(Artifact-Actor能力相关性). 在一个TRISO-Element中,*arf*所需的技术必须能够被*atrs*的能力满足,即$\forall TE=(arf,atv,atrs), arf.aftTechniques \subseteq \cup atr_i.Capabilities$ where $atr_i \in atrs$.

证明:假设该相关性不成立,即至少存在一种 *arf* 所需技术没有 *atrs* 掌握,那么显然,该 *arf* 无法被开发出来. □

定义 6. *Artifact*之间的前驱后继关系*PreceArf*(arf_a,arf_b)为

$\forall arf_a, arf_b, PreceArf(arf_a, arf_b)$ iff $\exists\{arf_a, atv_1, arf_1, \ldots, atv_n, arf_n, atv_{n+1}, arf_b | arf_i \in Artifacts, atv_i \in Activities, n \geq 0\}$

$arf_a \in atv_1.input \wedge arf_i \in atv_i.output \wedge arf_i \in atv_{i+1}.input \wedge arf_b \in atv_{i+1}.output\}$

定义 7. *Activity*之间的前驱后继关系*PreceAtv*(atv_a,atv_b)为

$\forall atv_a, atv_b, PreceArf(atv_a, atv_b)$ iff $\exists\{atv_a, arf_1, atv_1, \ldots, arf_n, atv_n, arf_{n+1}, atv_b | atv_i \in Activities, arf_i \in Artifacts, n \geq 0\}$

$arf_1 \in atv_a.output \wedge arf_i \in atv_i.input \wedge arf_{i+1} \in atv_i.output \wedge arf_{n+1} \in atv_b.input\}$

定理 5(Artifact-Activity 序列相关性). 在 TRISO-RM 中,*Activity* 之间的序列关系满足 *Artifact* 之间的序列关系,即

$\forall arf_a, arf_b \in Artifacts, atv_a, atv_b \in Activities$ where $(arf_a = atv_a.output \wedge arf_b = atv_b.output)$

if $PreceArf(arf_a, arf_b)$ then $PreceAtv(atv_a, atv_b)$.

证明:$\{atv_a, arf_a, atv_1, arf_1, \ldots, atv_n, arf_n, atv_b\}$即是定义 7 中所需序列,其中,$arf_a, atv_1, arf_1, \ldots, atv_n, arf_n$是由 $PreceArf(arf_a, arf_b)$得出的定义 6 中的序列.定理得证. □

如图 2 所示,由于这些 *Artifact*,*Activity*,*Actor* 之间的相关性(定理 1~定理 5,图中以虚线表示)的存在,TRISO-Element 中某一元素(的某些属性)的改变有可能会引起其他两个元素(相应属性)的改变,从而形成 TRISO-RM 中不同维度元素间的潜在关联.

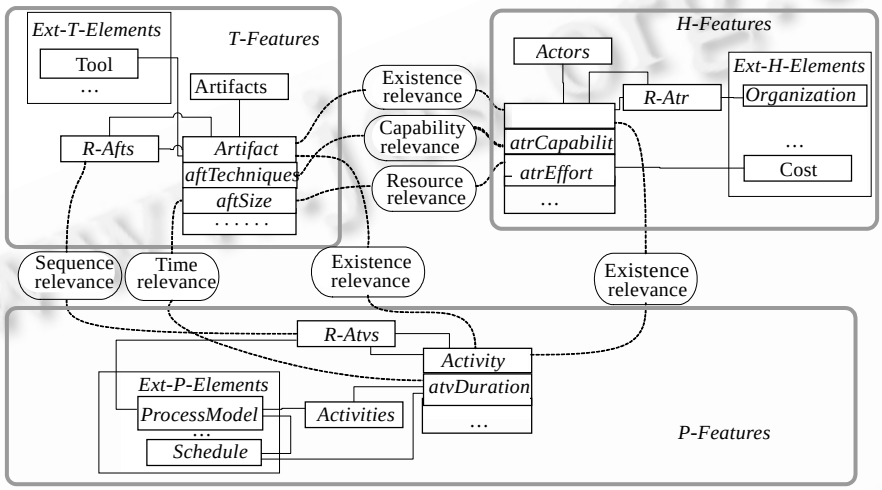

Fig.2 Inter-Dimensional interactions in TRISO-RM

图 2 TRISO-RM 三维之间的相关性

对这些关联的识别遵循相同的原理.例如:要分析在 *T-Features* 中的某一个属性 *W* 和 *H-Features* 中的某一个属性 *Q* 之间的关联,首先分析与 *W* 相关的 *Artifact* 集合以及如何相关,以确定所适用的相关性定理.如果与其存在性相关,则使用定理 1;若与技术、规模相关,则分别使用定理 4 和定理 2.通过相关性定理找到 *H-Features* 中对应的 *Actor* 集合 *A*,之后分析 *Q*,得到相关的 *Actor* 集合 *A′*.如果 *A* 和 *A′*之间有交集,则 *W* 和 *Q* 之间可能存在关联.

1.3 TRISO-RM的演化

TRISO-RM 描述了涉众对软件开发的各个方面的未来的期望及当前时间点之前这些期望的实现情况.软件开发始于 TRISO-RM 的初始状态,随着开发进程的推进,TRISO-RM 在不断发生演化,直至软件开发结束 TRISO-RM 中所描述的内容完全被实现.因此,软件开发过程同时也是 TRISO-RM 的演化过程.

2 TRISO-RM 对涉众协同的支持

TRISO-RM 提供了从涉众角度对软件开发进行描述的一个三维视图,并对其各维之间相互影响的描述和分析提供支持.从而,可以通过三维之间的关系对涉众的不同视角进行集成,支持涉众之间的协同.作为涉众共赢的前提条件,模型冲突的发现和消解是涉众协同中一个很重要的问题,也是 TRISO-RM 应用的一个重点.

2.1 模型冲突及其产生

根据文献[13]中的通用描述,我们在 TRISO-RM 中给出赢条件的定义.

定义 8(赢条件win condition). 一个赢条件是一个命题,描述了某涉众对TRISO-RM的技术、过程、人三维中某一维上部分或全部元素的某属性的期望.赢条件可以写作一个三元组(A,P,E),A是该赢条件针对的元素集合,P是所关心的属性,E则是对该属性的期望.例如,赢条件"总工作量不超过 300 人月"写作$(\{atr_i|atr_i\in Actors\}, atrEffort, totality<=300pm)$.

由于TRISO-RM中描述的维度之间的相互关联,在某一维上的赢条件经常(隐含地)导致对其他维度的某些期望.为了对这类期望进行讨论,我们有以下定义:

定义 9(暗含赢条件 implied win condition). 赢条件 WC 的推论称为暗含赢条件.

暗含赢条件也可看作是涉众的赢条件,这是因为:根据其定义,WC 暗含赢条件的失败将导致 WC 的失败.

定义 10(模型冲突model clash). 如果赢条件wc_1和wc_2可单独成立但不能同时成立,则称wc_1和wc_2之间产生模型冲突,写作$MClash(wc_1,wc_2)$.

按照文献[8]中的分类,模型冲突是一种冲突(conflict).但是,本文对其定义略有不同.在文献[8]中指出,冲突可能发生于多个需求之间.但是,对于多个赢条件之间的模型冲突,可以通过赢条件之间的"并"运算转化为定义10中定义的二元关系,尽管处理的复杂度变大,但其原理并无本质区别.

定理 6. 如果可单独成立的赢条件wc_1有暗含赢条件 wc_1',那么$MClash(wc_1',wc_2)\rightarrow MClash(wc_1,wc_2)$.

证明:对于 wc_1',wc_1,wc_2,因为$MClash(wc_1',wc_2)$,所以 wc_1'或wc_2不成立,且wc_2可单独成立.若wc_2不成立,则显然有$MClash(wc_1,wc_2)$;若 wc_1'不成立,由$wc_1\rightarrow wc_1'$得wc_1不成立,仍有$MClash(wc_1,wc_2)$. □

按照产生机理的不同,涉众的赢条件之间的冲突可以被分为两类:

(1) 显式冲突(explicit conflict):两个赢条件期望对同一维度元素属性进行赋值而导致的冲突.在这种情况中,冲突产生于元素和属性的指称域重叠的前提下,即冲突两个赢条件不仅所针对的元素集合有交集,而且是对同一类属性的期望.这种冲突的产生是直接的.例如:对于过程模型的取值,当顾客要求采用基于瀑布模型的软件过程,而开发人员希望采用基于迭代模型的极限编程过程时,显式冲突产生;

(2) 隐含冲突(implicit conflict):两个赢条件期望对不同维度元素属性进行赋值而导致的冲突.在这种情况下,是由于某个赢条件的暗含赢条件与另一个赢条件产生显式冲突导致的,因此是隐含的.例如,基于组件的开发(技术维)导致传统过程模型的失效(过程维)[24],因此与瀑布模型(过程维)发生冲突.

2.2 模型冲突的发现过程

在讨论模型冲突的发现过程之前,我们首先给出一种算法,用于发现赢条件在其他维度上的暗含赢条件.限于篇幅,在算法中只给出了从技术维的赢条件发现其在人维上的暗含赢条件的过程,其余 5 种方向(人→技术、过程→人、人→过程、技术→过程、过程→技术)的过程与此类似,其区别仅在于所用相关性定理不同,这里从略.由于存在性对所关注元素的所有属性都有影响,因此,当涉及到存在性的时候,对所针对元素的各方面都需要进行考察.

算法 1. 暗含赢条件的发现算法 $ImpliedWCFinder(wc,D,D')$.

输入:(1) 待检查的某一维上赢条件 $wc=(A,P,E)$;

(2) wc 所在的维度 D;

(3) wc 的暗含赢条件所在的维度 D'.

输出:赢条件集合$\{wc_1,wc_2,\ldots,wc_n\}$是$wc$在$D'$维度上的暗含赢条件集合.

```
0.  Begin
1.      If ((D==T-Features) && (D'==H-Features))
2.          根据 A 中各元素所在的 TRISO-Element,得到所对应的 Actor 集合 A'
3.          If P 关于 A 的存在性 Then 使用定理 1、定理 2、定理 4,得到
```

4. return {wc_1=(A',存在性,E_1),wc_2=(A',*atrEffort*,E_2),wc_3=(A',*atrCapabilities*,E_3)}
5. // E_1,E_2,E_3,E_4,E_5的取值视E的具体情况而定
6. EndIf
7. If P 关于 A 的 *arfSize* Then 使用定理 2 得到
8. return {wc_4=(A',*atrEffort*,E_4)}
9. EndIf
10. If P 关于 A 的 *arfTechniques* Then 使用定理 4 得到
11. return {wc_5=(A',*atrCapabilities*,E_5)}
12. EndIf
13. EndIf
14. If ((D==技术维) && (D'==人维)) //以下略
15. … …
16. End

模型冲突的分析和发现过程如下:

步骤 1. 获取涉众的赢条件.

完整地发现涉众的赢条件,是对其进行完整检查的基础.较为常用的方法包括群体讨论如头脑风暴方法,或者通过基于经验知识库的工具支持.TRISO-RM 的三维结构为赢条件的获得提供了一个基本的框架,也即从技术、过程、人/经济 3 个方面进行识别.这一步的输出是所有的赢条件的描述及其所有者.

涉众的赢条件往往是用自然语言表述的,并且有可能并非直接以定义 8 中的三元组形式.因此,在获取赢条件之后,需要进行初步的分析,并写成这种三元组形式.

步骤 2. 显式冲突的检查.

显式冲突检查的重点在于发现相关的赢条件,即作用于相同元素的相同属性上的赢条件.在此之后,冲突的发现方法应当根据具体情况来选择,常用的方法如人工审查、逻辑检查等.由于这不是本文所讨论的重点,所以在算法 1 中,我们以函数 ClashCheck 对其进行抽象表述.需要注意的是:在开发的初始阶段,由于赢条件经常是以自然语言描述,这种检查应当使相关的涉众参与进来,以消除由于理解的偏差导致的错误.这里给出的算法在发现第 1 个冲突后即输出并停止,也可修改为直接检查所有的冲突.

算法 2. 显式冲突的检查算法 *ExplicitClashFinder*.

输入:待检查的某一维上赢条件集合{wc_1,wc_2,…,wc_n},其中,wc_i=(A_i,P_i,E_i),i=1,...,n;

输出:发现的第 1 个模型冲突 *mc*.

0. Begin
1. For i=1 to n–1
2. For j=i+1 to n
3. If ((A_i∩A_j≠∅) && (P_i∩P_j≠∅)) Then //发现元素和属性的指称域重叠
4. If (*ClashCheck*(wc_i,wc_j)==true) Then
5. Return *mc*=*MClash*(wc_i,wc_j); //发现模型冲突
6. EndIf
7. EndIf
8. EndFor
9. EndFor
10. End.

步骤 3. 隐含冲突的检查.

在 TRISO-RM 的支持下,发现隐含模型冲突的基本过程在于:通过各维之间的相关性定理,得到每个赢条件

在 TRISO-RM 中各个维度上的暗含赢条件;从而,将隐含冲突的检查转化为显式冲突的检查.这里给出的算法,在发现第 1 个冲突后即输出并停止.

算法 3. 隐含冲突的检查算法 *ImplicitClashFinder*(以 *T-Features* 维和 *P-Features* 维之间的冲突为例).

输入:待检查的

(1) *T-Features*维上的赢条件$WC_t=\{wc_{t1},\ldots,wc_{tn}\}$,其中,$wc_{ti}=(A_{ti},P_{ti},E_{ti}),i=1,...,n$;

(2) *P-Features*维上的赢条件$WC_p=\{wc_{p1},\ldots,wc_{pm}\}$,其中,$wc_{pj}=(A_{pj},P_{pj},E_{pj}),j=1,...,m$.

输出:发现的第 1 个模型冲突 *mc*.

```
0.  Begin
1.      For i=1 to n
2.          WC_ti=ImpliedWCFinder(wc_ti,T-Features,P-Features)   //过程维上的暗含赢条件集合
3.          temp_mc=ExplicitClashFinder(WC_ti∪WC_p);   //检查WC_ti与过程维上赢条件的显式冲突
4.          if temp_mc≠∅   //发现冲突
5.              Return mc=temp_mc
6.      EndFor
7.      For i=1 to m
8.          WC_pi=ImpliedWCFinder(wc_pi,P-Features,T-Features)   //技术维上的暗含赢条件集合
9.          temp_mc=ExplicitClashFinder(WC_pi∪WC_t);   //检查WC_pi与技术维上赢条件的显式冲突
10.         if temp_mc≠∅   //发现冲突
11.             Return mc=temp_mc
12.     EndFor
13. End.
```

需要说明的是:在某两个维度的赢条件之间的隐含冲突检查不是对称的,因此,两个方向上都需要进行.这是因为暗含赢条件是原赢条件的推论,而非等价条件.

2.3 模型冲突的处理

发现模型冲突之后,需要通过涉众之间的协商,由某一方或多方做出让步(tradeoff)以消除这些冲突,从而为共赢奠定基础.为了提高协商的效率和效果,需要支持原始冲突条件的识别、协商参与者的识别、协商方案的影响分析及新冲突的避免,并对协商过程进行支持.由于该部分内容非本文重点,因此这里仅进行简要说明.

当冲突被发现以后,首先需要回溯至涉众最初所提出的赢条件,我们将其称作原始冲突条件.在算法 2 和算法 3 的输出中,都给出了原始冲突条件.原始冲突条件的所有者都是协商的参加者.由于用户是原始需求的拥有者,而三维上发生的协商结果都可能对其造成潜在的影响,因此,用户代表应该参加所有的协商活动.

在协商中,不适当的让步方案会导致新的模型冲突的产生.一类典型例子就是不适当的需求变更.例如:在基于瀑布模型的过程中,开发人员被要求接受用户的需求变更请求以消除某个模型冲突.由于视角的局限性,开发人员在进行分析时,如果仅考虑技术上的可行性,可能会接受这一变更请求.但是,由于通过 TRISO-RM 对这一让步方案对过程或者进一步对预算的影响进行分析时,可以看到这一变更可能会带来不可接受的成本超支,从而导致顾客在成本方面的赢条件失败.因此,在进行让步方案分析时,需要通过 TRISO-RM 中各部分之间的关系,分析其带来的连锁影响,尽量保证没有新的模型冲突发生.如果新的模型冲突的发生是不可避免的,则进行进一步的协商,直至所有的冲突都被消除.

2.4 赢条件及模型冲突在TRISO-RM演化中的监控

在 TRISO-RM 的演化过程中,随着开发活动的进展,涉众的赢条件也在逐渐发生变化.典型的变化包括:新的涉众的加入;涉众发现和提出新的赢条件;涉众对原有的赢条件进行调整.而随着赢条件的变化,新的模型冲突往往也会继续出现.因此,在 TRISO-RM 的演化过程中,对于赢条件和模型冲突的监控应该并行持续进行,与

项目的跟踪同步.这种监控主要在两个方面实行:

1) 监控由涉众提出的赢条件的调整.需要注意的是,这里的支持包括两个方面:一是通过 TRISO-RM 向涉众提供充分的信息,以支持涉众对软件开发各维度进展的了解,从而更及时地调整其赢条件;二是在此基础上及时获取涉众对赢条件的调整;
2) 当赢条件发生变化时,对模型冲突进行监控.即通过第 2.2 节中的方法进行模型冲突的发现,使用第 2.3 节中讨论的过程对冲突进行处理.

在现有软件开发环境下,可以通过项目跟踪,使软件过程在活动的粒度上得以监控;对制品的监控通过配置管理进行;人力资源、成本等也分别可以通过PSP[25]或者基于价值的方法[26]等进行管理.

3 实例研究

为了对 TRISO-RM 方法进行验证,我们在大型软件系统 SoftPM 的开发中对其进行了应用.在这一项目中,包括用户、开发团队、顾客等多方面的涉众参与其开发过程,而分布于不同的组织中,分别对成本、技术、使用等方面负责,具有典型的视角的局部性.在 SoftPM 中的实践证明:TRISO-RM 对于涉众的协同,尤其是模型冲突的分析和发现,提供了有效的支持.

3.1 项目概况

SoftPM 是受国家 863 计划支持的一个软件过程管理系统,其目的是为了支持软件过程改进,帮助企业实现符合 CMM/CMMI 标准的管理体系.在这个项目中,项目周期和预算由基金管理部门确定,具体的系统需求需要从目标用户(软件产业孵化器的企业)中获取.为开发这一系统,中国科学院软件研究所(ISCAS)组建了一支开发团队 SoftPMGroup.若干从事实际项目开发工作的团队 TeamA 也参与进来,作为用户代表.在开始阶段中,识别出 5 类涉众,见表 1.

Table 1 Stakeholders in SoftPM project

表 1 SoftPM 中的涉众

Categories	Representative (s)
Customer	Government
User	TeamA
Senior manager (SM)	ISCAS administrator
Project manager (PM)	SoftPMGroup leader
Developer	SoftPMGroup members

3.2 TRISO-RM的初步建立

SoftPM 的启动过程就是 TRISO-RM 的初步建立过程.在该阶段,各部分都还没有经过细化,原始需求即为开发 SoftPM,其具体功能等需求仍有待进一步的获取;所产生的 *Artifact* 就是 SoftPM 平台;所执行的活动就是 SoftPM 的开发活动,其参与者为所有 *Actors*,输出为 SoftPM.限于篇幅,具体数据从略.

3.3 模型冲突的识别与消除

识别赢条件

从涉众处获得的部分赢条件示例见表 2.

在实践中,完整的获取涉众的赢条件是比较困难的,因为涉众经常将某些赢条件作为当然假设,而没有意识到其他可能性.在 SoftPM 项目中,我们采用了一个经验知识库(如图 3 所示)作为支持,其中记录了在本领域历史项目中各类涉众所关注的赢条件.由于项目刚刚启动,这些赢条件的描述多是针对各维的整体特性.在 SoftPM 项目中,顾客(政府)所关注的是一些整体性的目标,例如整个项目的工期、成本以及是否能够满足用户的要求.由于几乎没有类似的商业化的系统,因此,用户的需求还很模糊.项目的高层经理希望项目开发过程本身能够采用较易符合 CMM 标准的瀑布过程模型.项目经理希望开发过程不被其他活动打断,例如培训课程.开发人员希望能使用可重用的组件,例如商业构件或者开源组件,从而可以减少工作量.开发人员还希望能够采用极限编程

的方法.

Table 2 Win conditions found at the beginning

表 2 初始阶段获取的赢条件

ID	Description	Owner
wc_1	Budget is XXX,XXX yuan	Customer
wc_2	Milestones are pre-defined	Customer
wc_3	SoftPM is compatible with existing systems	User
wc_4	IKIWISI (i know it when i see it)	User
wc_5	Use waterfall development process	SM
wc_6	Developers focus on development activities	PM
wc_7	Use COTS (commercial-off-the—shelf)	Developer
wc_8	Use XP (eXtreme programming)	Developer
wc_9	Lack the knowledge and skills on COTS	Developer
wc	Lack the knowledge on existing systems	Developer

在获得赢条件之后,为将其表述成定义 8 中的三元组形式,需要进一步的分析.有时候,这种转换不是直接的,例如wc_4,实际上表述了用户希望能在可运行系统的基础上提出进一步的需求,因此,可以表述为对制品之间的序列关系的期望.有时候,一个赢条件实际上涵盖了多个期望,如wc_8中的极限编程方法既包括过程维期望,又包括技术维的考虑.这种情况下,应将其拆分成一系列子赢条件.限于篇幅,这部分表述从略,部分内容在表 3 中列出.

Table 3 Model clash analysis using algorithm 1 & algorithm 2

表 3 使用算法 1 和算法 2 发现模型冲突

Dimension	Win conditions and implied win conditions	Clashes	
		Explicit	Implicit
T-Features	[wc_3] *ExistingSystem*∈*Artifact.Techiniques*	/	/
	[wc_4] $\exists arf_1,arf_2[arf_1$="Design"$\wedge arf_2$="ReqAnalysis"$\wedge$*PreceArf*$(arf_1,arf_2)]$	/	/
	[wc_7] *Cots*∈*Artifact.Techiniques*$\wedge\exists arf_3,arf_4[arf_3$="Code"$\wedge arf_2$="Design"$\wedge$ *PreceArf*$(arf_3,arf_4)]$	/	/
P-Features	[wc_5] *P-Features.Processmodel=waterfallmodel*	/	/
	[wc_{51}] $\neg\exists atv_1,atv_2[atv_1$="Design"$\wedge atv_2$="ReqAnalysis"$\wedge$*PreceAtv*$(atv_1,atv_2)]$	/	/
	[wc_{52}] $\neg\exists atv_3,atv_4[atv_3$="Code"$\wedge atv_2$="Design"$\wedge$*PreceAtv*$(atv_3,atv_4)]$	/	/
	[wc_6] $\neg\exists atv$∈*Activities*[*atv.Type*="NonDevelop"]	/	/
	[wc_8] *P-Features.Processmodel=iterativemodel*	[wc_5]	/
	[$wc_{4\text{-}1}$] $\exists atv_1,atv_2[atv_1$="Design"$\wedge atv_2$="ReqAnalysis"$\wedge$*PreceAtv*$(atv_1,atv_2)]$	/	[wc_{51}]
	[$wc_{7\text{-}1}$] $\exists atv_3,atv_4[atv_3$="Code"$\wedge atv_2$="Design"$\wedge$*PreceAtv*$(atv_3,atv_4)]$	/	[wc_{52}]
H-Features	[wc_9] *Cots*∉*Actor.Capabilities*	/	/
	[wc_A] *ExistingSystem*∉*Actor.Capabilities*	/	/
	[$wc_{3\text{-}1}$] *ExistingSystem*∈*Actor.Capabilities*	/	[wc_A]
	[$wc_{7\text{-}2}$] *Cots*∈*Actor.Capabilities*	/	[wc_9]

发现与消除冲突

根据第 2.2 节中的 3 种算法,我们发现了一系列的显式冲突和隐含冲突,见表 3.需要说明的是:涉众的赢条件往往是以自然语言的形式提出来的,因此,其对于各维的赋值或约束经常需要进一步的分析和确认.这一分析的过程,其目的不仅在于发现已有的赢条件中的冲突,更为根本的目的在于确保任何一个涉众的赢条件被所有其他涉众所接受.因此,分析的过程同时也是一个对原有赢条件集合进行更新的过程.在本例中,wc_9和wc_A即是为了检查wc_3和wc_7的可能性而继续从开发人员处获得的赢条件.

在表 3 中,我们在最后两列标出了发现的显式和隐含冲突,对于每个冲突,仅在发现的地方标注 1 次.其中:wc_5和wc_8之间是显式冲突;另外 4 组是隐含冲突.与第 2.2 节中的算法稍有不同的是:为了表示方便,这里将所有冲突发现以后才停止算法的执行,并且采用$wc_{i\text{-}j}$来表示从赢条件wc_i得到的第j个隐含赢条件,而用wc_{ij}表示赢条件wc_i的第j个经过进一步分解得到的子赢条件.同时,为了理解的方便,将三元组的内容合在一起来表述.

根据表 2 和表 3,导致这些冲突的原始冲突条件及其所有者可以很容易地识别出来.在消除冲突的协商中,我们使用TRISO-RM向各相关涉众展示了其所提出的赢条件对其他涉众造成的影响.同样地,在涉众提出让步方案的时候,我们对协商方案的影响也进行了类似的分析,以避免产生新的模型冲突.在本例中,当涉众决定使

用COTS并放弃瀑布模型、对开发人员进行培训以解决与wc_5和wc_9的冲突时,新引入的培训活动又将与wc_6产生新的冲突,因为项目经理不希望开发团队从事除开发活动之外的活动.

图 3 所示的经验知识库在模型冲突的发现与消除中也起到了重要的作用.这是因为:不论是赢条件的分解还是其在各个维上推论的获得,都需要涉众进行反馈和确认,难以被完全自动执行,而可以通过经验知识的启发提高其准确度和完整性.当冲突发生后进行的协商中,以往开发实践中的类似解决方案也具有参考作用.

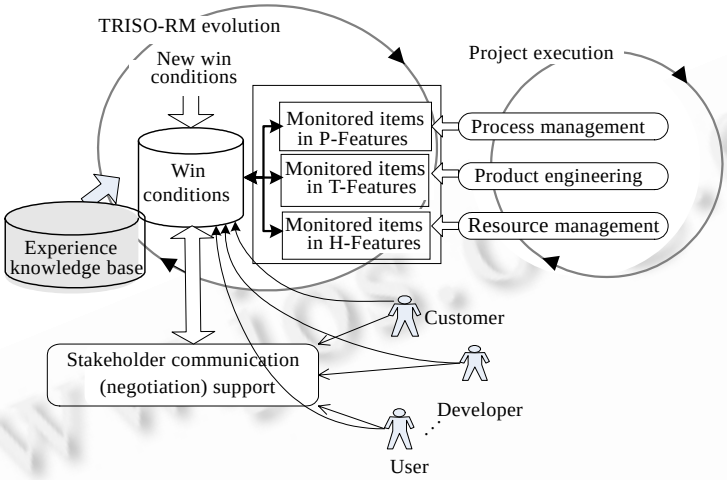

Fig.3 Monitoring TRISO-RM evolution

图 3 对 TRISO-RM 演化的监控

3.4 对TRISO-RM演化的持续监控

在完成开始阶段的模型冲突检查之后,初始的赢条件、相关的元素及其状态都在 TRISO-RM 中得到了描述.在开发进程中,随着分别以制品、活动、参与者为核心的技术、过程、人三维的演化,赢条件也在持续演化.对 TRISO-RM 的演化的持续监控主要有两个目的:向涉众提供关于软件开发进展的丰富视图以便及时发现赢条件的变化,并在此基础上支持对模型冲突的持续监控.

如图 3 所示,在 SoftPM 的开发中,我们使用了 3 个(抽象的)平台以支持项目实施并收集 TRISO-RM 的技术、过程、人三维上的信息.其中既包括自动化工具支持,也包括一些人工的数据统计.

- 产品工程平台:提供与制品相关因素的管理,对应 TRISO-RM 中技术维的 *T-Features* 中相关数据的收集.一些典型的组成部分包括进行配置管理的 CVS 系统、进行测试管理的工具等;
- 过程管理平台:提供与活动相关因素的管理,对应 TRISO-RM 中过程维的 *P-Features* 相关数据的收集.我们采用一个功能类似于微软的 Project 的工具对活动进行计划、跟踪、数据统计等;
- 资源管理平台:提供与参与者相关因素的管理,对应 TRISO-RM 中人维的 *H-Features* 相关数据的收集.COCOMOII 被用于工作量和成本的估算.该平台中还包括人力资源的管理、培训等支持.

我们为每一个赢条件预定义了一组“监控项”,它们的状态变化与赢条件相关.通过对这些监控项的监控来跟踪赢条件的变化.典型的监控项包括活动进展状况、可用资源数量等.当监控项发生变化时,涉众可以对原有赢条件进行检查,以确定是否进行修改.涉众也可以(主动)修改赢条件,或提出新的赢条件.一旦一个新的赢条件被放入赢条件集合中,则首先对其进行模型冲突的检查和处理,并建立起相应的监控项.

当冲突被发现时,相关的涉众通过会议或电子邮件等方式进行协商以消除冲突,并使用 TRISO-RM 避免新的冲突产生.

通过 TRISO-RM 方法,不同涉众的赢条件得到了很好的满足,SoftPM 项目在限定的工期内以合理的成本开发出满足用户需求的高质量的软件系统,并实现了很多可复用的构件.关于该平台的更详细信息参见文献[21].

4 结束语

在多涉众参与的软件开发中,项目成功的充分必要条件是涉众达到共赢状态,而及时地发现和消除赢条件之间的模型冲突是实现共赢的前提条件.在目前已有的研究中,或仅针对软件需求[2,3,8,9],或关注于冲突发现和解决的抽象框架而缺乏具体性[6,15−17],或依赖于经验[1]而难于学习和重复.尤其对于不同关注方面的赢条件之间的相互影响机制缺乏系统的解释,难以有效发现它们之间的冲突.在较具代表性的MBASE(model-based system architecting and software engineering,基于模型的系统架构与软件工程)[20]方法中,将赢条件划分为 4 类模型,对各类赢条件之间的关系给出了基于经验的说明.但是,由于MBASE并没有对模型冲突发生的机制进行深入的分析和解释,对不同类别赢条件之间的相互影响也缺乏具有普遍性的分析方法.

本文提出的三维需求模型通过将软件开发中的关键因素划分为技术、过程、人三维,并通过三维的核心元素 *Artifact*,*Activity*,*Actor* 之间的依存关系和相关性规则,分析和建立三维上各因素之间的相互影响.从而,各维的赢条件——也即对这些因素的期望——之间的相互影响也得到了建立.在此基础上,本文提出了发现和消除模型冲突的方法及过程,并通过在大型软件系统 SoftPM 的开发过程中的成功应用对其有效性进行了验证.

在本文所介绍的冲突发现算法中,要求使用者对于TRISO-RM的三维上的要素进行详尽分析,在实际应用中效率有待提高.有两种可能的改善方法:第 1 种方法是对执行过程进行自动化的支持,这就要求提高TRISO-RM的形式化程度;第 2 种方法是采用诸如冲突模式或者启发式规则的形式[8],在较高的抽象层次对冲突进行分析,但是这样有可能对分析的准确性和完整性带来影响.另外,如何利用TRISO-RM来支持涉众协商过程[13,14]从而进行冲突消解、如何对TRISO-RM的应用提供进一步的工具支持等,也是需要深入探讨的问题.对这些内容进行进一步的探索,是我们下一步要进行的工作.

References:

[1] Boehm B, Ross R. Theory-W software project management: Principles and examples. IEEE Trans. on Software Engineering, 1989, 15(7):902−916.

[2] Heitmeyer CL, Jeffords RD, Labaw BG. Automated consistency checking of requirements specifications. ACM Trans. on Software Engineering and Methodology, 1996,5(3):231−261.

[3] Chan W, Anderson RJ, Beame P, Burns S, Modugno F, Notkin D, Reese JD. Model checking large software specifications. IEEE Trans. on Software Engineering, 1998,24(7):498−520.

[4] Nuseibeh B, Kramer J, Finkelstein A. A framework for expressing the relationships between multiple views in requirements specification. IEEE Trans. on Software Engineering, 1994,20(10):760−773.

[5] Kotonya G, Sommerville I. Requirements engineering with viewpoints. Software Engineering Journal, 1996,11(1):5−18.

[6] Dardenne A, Lamsweerde V, Fickas S. Goal-Directed requirements acquisition. Science of Computer Programming, 1993,20(1-2): 3−50.

[7] Yu E. Modeling strategic relationships for process reengineering [Ph.D. Thesis]. Toronto: University of Toronto, 1995.

[8] Lamsweerde V, Letier E, Darimont R. Managing conflicts in goal-driven requirements engineering. IEEE Trans. on Software Engineering, 1998,24(11):908−926.

[9] Rashid A, Moreira A, Araújo J. Modularisation and composition of aspectual requirements. In: Proc. of the 2nd Int'l Conf. on Aspect-Oriented Software Development. New York: ACM Press, 2003. 11−20. http://portal.acm.org/toc.cfm?id=643603&type=proceeding&coll=ACM&dl=ACM&CFID=33826384&CFTOKEN=16151607

[10] Rashid A, Sawyer P, Moreira A, Araújo J. Early aspects: A model for aspect-oriented requirements engineering. In: Proc. of the 10th Anniversary IEEE Joint Int'l Conf. on Requirements Engineering. Los Alamitos: IEEE Computer Society Press, 2002. 199−202. http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=22462&isYear=2002

[11] Zhu XF, Jin Z. Managing the inconsistency of software requirements. Journal of Software, 2005,16(7):1221−1231 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1221.htm

[12] Spanoudakis G, Zisman A. Inconsistency management in software engineering: Survey and open research issues. In: Chang SK, ed. Handbook of Software Engineering and Knowledge Engineering. Singapore: World Scientific Publishing Co., 2001. 329−380.

[13] Boehm B, Bose P, Horowitz E, Lee MJ. Requirements negotiation and renegotiation aids: A theory-w based spiral approach. In: Proc. of the 17th Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society Press, 1995. 243−254. http://portal.acm.org/citation.cfm?id=225037&coll=ACM&dl=ACM&CFID=33826384&CFTOKEN=16151607

[14] Boehm B, Grünbacher P, Briggs RO. Developing groupware for requirements negotiation: Lessons learned. IEEE Software, 2001, 18(3):46−55.

[15] Nuseibeh B, Easterbrook S, Russo A. Leveraging inconsistency in software development. Computer, 2000,33(4):24−29.

[16] Robinson WN, Volkov V. Supporting the negotiation life cycle. Communications of the ACM, 1998,41(5):95−102.

[17] Spanoudakis G, Finkelstein A, Till D. Overlaps in requirements engineering. Automated Software Engineering, 1999,6(2): 171−198.

[18] Robinson WN, Fickas S. Supporting multiple perspective requirements engineering. In: Lawrence B, ed. Proc. of the 1st Int'l Conf. on Requirements Engineering. Colorado Springs: IEEE Computer Society Press, 1994. 206−215.

[19] Boehm B, Port D. When models collide: Lessons from software systems analysis. IT Professional, 1999,1(1):49−56.

[20] Boehm B, Port D. Escaping the software tar pit: Model clashes and how to avoid them. ACM SIGSOFT Software Engineering Notes, 1999,24(1):36−48.

[21] Wang Q, Li M. Software process management: Practices in China. In: Li M, Boehm B, Osterweil LJ, eds. Unifying the Software Process Spectrum. LNCS 3840, Berlin: Springer-Verlag, 2005. 317−331.

[22] Li M. Expanding the horizons of software development processes: A 3-D integrated methodology. In: Li M, Boehm B, Osterweil LJ, eds. Unifying the Software Process Spectrum. LNCS 3840, Berlin: Springer-Verlag, 2005. 54−67.

[23] Boehm B, Horowitz E, Madachy R, Reifer D, Clark BK, Steece B, Brown AW, Chulani S, Abts C. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

[24] Yang Y, Bhuta J, Boehm B, Port D. Value-Based processes for COTS-based applications. IEEE Software, 2005,22(4):54−62.

[25] Humphrey W S. Introduction to the Personal Software Process (Gravure). Beijing: Posts & Telecommunications Press, 2002.

[26] Boehm B, Huang LG. Value-Based software engineering: A case study. Computer, 2003,36(3):33−41.

附中文参考文献:

[11] 朱雪峰,金芝.关于软件需求中的不一致性管理.软件学报,2005,16(7):1221−1231. http://www.jos.org.cn/1000-9825/16/1221.htm

王继喆(1978－),男,山东临沂人,博士生,主要研究领域为软件需求工程与协同技术.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件过程技术与需求工程.