基于多任务优先算法的异地敏捷软件开发任务分派①

殷 茗, 马 静

(西北工业大学 软件与微电子学院, 西安 710072)

摘 要: 针对日益受欢迎的异地敏捷软件开发, 提出了一种基于多任务优先算法的任务分派方法, 并运用数学计算方法进行任务分派. 通过多模型调查研究, 较全面综合考虑异地敏捷开发中多方面影响因素, 给出一种具体分派方法, 以提高任务分派的有效性. 经验证, 该任务分派方法适合异地敏捷软件开发.

关键词: 异地分布式; 敏捷开发; 任务协调; 任务分派

Task Assignment of Geographically Distributed Agile Software Development Based on the Multi-Task Priority Algorithm

YIN Ming, MA Jing

(Institute of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: For distributed agile software development becoming increasingly popular, this study seeks to shed light on proposing a task assignment method based on the multi-task priority algorithm, which is using mathematical calculations to assign tasks. According to multi-model researches, more comprehensive factors of distributed agile software development were considered. A specific assignment method was given, to improve the effectiveness of task assignment. Through verification, the task assignment method is suitable for distributed agile software development.

Key words: geographically distributed; agile software development; task coordination; task assignment

随着全球化发展和由此引发的异地软件开发, 理解并运用工程和管理的方法显得尤为重要[1]. 异地分布式开发是一个令人矛盾的过程: 一方面, 采用异地开发可以节省劳动力成本, 充分利用全球范围内的智力资源; 另一方面, 除了至今仍未完全解决的任务协调改善技术, 及时沟通与交流也存在重大问题. 这些问题会导致知识传播困难[2], 甚至还存在智力资源保护问题[3]. 鉴于以上优势和劣势, 任务分派对项目的成败至关重要, 是每个异地分布式团队都要考虑的问题. 软件开发中任务分派可以定义为: 考虑到现有限制, 决定以何种方式执行项目任务, 如何分配现有资源的过程[4]. 定义分布式项目的任务往往涉及到与任务相关的技术限制, 如团队成员经验及编程能力、语言沟通、文化差异等, 同时还需要考虑到项目本身复杂度、依赖关系. 所考虑到项目影响因素不同, 会直接影响到任务分派结果, 从而影响项目进展. 因此, 需要一个系统的方法来进行任务分派. 对于任何一个软件开发团队来说, 任务的协调与分派是一个非常困难的问题, 这是由异地软件开发活动的性质及其内部关系决定的[5-7]. 一方面, 分布式的任务分派需要考虑的因素非常多, 如地域、开发者各方面的素质与能力等; 另一方面, 企业的任务分派内容及结构随着市场竞争及战略发展目标的变化而需要随时调整[8]. 敏捷开发团队需要频繁、非正式的交流, 如每日站立会议, 这对异地分布式开发团队来说尤其困难, 更加需要优化分派策略. 此外, 异地分布式开发的动态性使其任务分派与其他行业相比有很大不同. 近年来, 敏捷软件开发方法迅速发展并受到广泛欢迎, 这些开发方法强调

① 基金项目:国家自然科学基金(71201124,71172124,71102087);国家留学基金(201203070070);高等学校博士学科点专项科研基金(20116102110036);西安市科技计划(CXY10102)

收稿时间:2014-04-22;收到修改稿时间:2014-05-12

降低对前期计划的依赖, 利用人工方法增强协调能力, 它们强调原子化方法和流水线技术. 但当一个开发团队拥有 10 人以上开发成员或者开发成员分布在不同地区, 团队协作和任务分派中将存在较大问题.

目前, IT 人士对各领域的任务分派研究日益增多. 早期, 国内学者对三种典型的任务分派算法进行了分析[9], 基于图论的分配算法仅适用于处理机小于 3 的环境中, 局限性较大; 整体规划方法的算法复杂度比较大, 并随问题规模呈指数增长, 限制了算法的实用性; 试探法中的最优解通常很难获得, 仍有一些方面未考虑到, 难以满足实际问题的需要. 随之, 基于多准则的任务分派算法[10], 提出了在工作流管理系统中使任务分配更加人性化、员工工作效率更高的动态分派方法. 近来, 针对作业车间调度和任务分派问题, 以制造系统的生产成本、准时交货为目标[11], 提出了两阶段蚁群算法的带非等效并行机的作业车间调度. 以上研究并非直接应用于异地软件开发中的任务分配, 但分配思想具有相通之处. 国外学者关于异地软件开发中任务分派的研究相对较多. 基于模型的分派方法提出了三种模型[12]: 集成风险模型, 指出每一个任务分派过程中存在的风险; 优化模型, 利用贝叶斯网络就多个影响因素提出分配方案; 精力开销模型, 对每次分派任务进行估计. 多标准分派模型改进了 TAMRI 模型[13],结合贝叶斯网络, 提出了一种任务分派方法并通过软件进行了验证. 许多任务决策者在进行任务分派时往往根据经验或者临时决定, Marques A B 等提出了本体论[14], 提醒项目经理应该综合考虑多方面影响因素, 并为研研究人员提供了一个框架来定义流程和设计工具来支持这样的活动. 研究中大多仅提出了任务分派的理论模型或者对任务分派的内容进行了详细定义, 真正为实际任务分派尤其是异地软件开发任务分派的具体实施提供方法与步骤的研究还比较少. 针对该方面研究的不足, 提出了一种关于异地敏捷开发的任务分派方法, 给出了具体建模方法和计算步骤, 为软件开发的任务分派提供了可以借鉴的分配方法.

1 软件开发中的任务分派

关于解决团队任务协调与分派问题, 提出了很多方法, 这种新兴的领域产生了许多解决问题的方案. Shim, Lee, et al 提出了另一个基于模型的方法, 使建模的任务分配给策略[15]. 具体表现为利用基于 UML 元模型描述了任务分配的四个关键要素: 组织, 流程, 工作产品和项目. 模型包含的信息的角色, 工作产品, 它们之间的关系, 以及为每个团队成员的工作量信息, 同时还把定义项目的任务分配策略与约束结合起来. 由于这个模型包含了几项最简单最基本的元素, 因此能够适用于不同的开发环境, 但是没有考虑到工作产品之间的关系, 把每一个任务作为一个单独的实例. Shen, Tzeng, et al.提出了多标准工作流管理系统内的任务分配模型[16]. 该模型描述定性措施, 如个人能力, 由模糊数来量化使用的语言尺度. 团队成员对任务的适用性使用四个标准来评估: 能力、工作量、任务的相似性及团队成员之间的关系. 对每一个任务, 其适宜得分是通过聚集团队中所有合资格团队成员中两两比较的结果. 如果一个任务必须由某个团队执行, 则得到社会关系分数. 这个模型很灵活并且容易理解, 但是由于元模型为基础的方法, 该模型假设任务之间彼此独立. Duggan, Byrne, and Lyons 提出在软件开发实施阶段任务分配的优化方法[17], 该方法利用多目标演进算法和遗传算法. 他们提出一个软件工程项目有许多目标, 这些目标之间往往会有冲突. 多目标演进算法帮助项目经理平衡各方面的目标, 为单个团队成员产生一组优化调度. 虽然这种方法估计成本和进度非常有效, 但该方法把任务的协调和分配均视为静态化问题, 没有考虑到紧急变更, 在正在执行的项目中进行任务分配时将产生较多问题. Bikram Sengupta 等提出了使异地软件开发更有效的四个领域需要解决的问题[18]:协作软件工具、知识获取与管理、分布式设置中的测试和过程及度量问题, 在理论上给出了异地软件开发中的重要相关问题, 妥善解决以上问题理论上能提高软件开发效率. 以上研究表明, 软件开发流程模型中存在相互关联的活动, 需要一种模型或方法, 适用于任务之间相互联系的、动态的任务分配方法, 并具体应用于实际开发过程中. 综上所述, 异地敏捷开发面临两方面的问题: 给定任务的优先级; 确定最佳人员执行任务.

2 异地分布式开发任务分派模型

2.1 任务分派层次模型

图 1 为任务分派模型的层次结构图, 指出了分布

开发任务的组成.

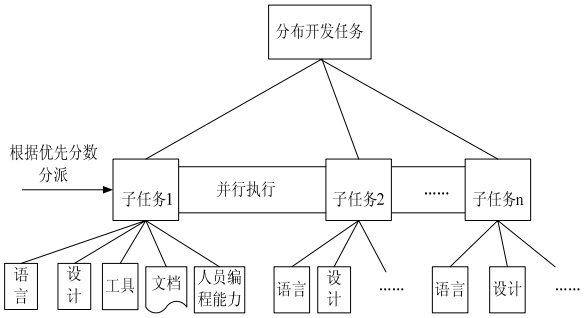

图 1　任务分派的层次结构图

在这种层次结构下, 异地软件开发将开发任务分为若干个子任务. 每一个子任务可以看作一个单独的开发团队, 彼此开发并行模块. 项目经理根据项目的时间要求和任务量大小, 对项目开发活动进行规划, 根据子团队中任务的关键程度、人员的编程能力及分布式地区优势, 将分布开发的任务分派给不同的子团队, 子团队的开发人员可以在不同的工作地区、地域并行进行, 但他们之间并不是完全独立的, 需要不断沟通协调工作; 他们也可以独立收集一些经常变化的分布式需求, 汇总在一起, 最终致力于项目的成功. 在软件开发过程中, 项目经理实时了解项目进展情况, 对任务的执行进行监控, 在特殊情况下项目经理可以提升任务的优先级, 以保证项目顺利完成. 该分派层次结构的显著特点是多个子任务可以并行执行, 在一定时期内互不影响, 提高了软件开发效率; 每个子团队软件开发、迭代开发中计算影响因素的方式完全相同. 由于决策因素相同(语言、设计、工具、文档、人员编程能力), 故决策结果相对可靠. 语言的选择也将影响软件开发进度和效率. Groovy 是 Java 平台上设计的面向对象编程语言, 这门动态语言拥有类似 Python、Ruby、Small Talk 中的一些语言特性, 也可以作为 Java 平台上的脚本语言使用; 支持 DSL 和其他简洁语法, 使代码变得易于阅读和维护; 受检查的异常类型可以不用捕获; 在开发 Web、GUI、 数据库和控制台程序时, 减少了框架性代码, 大大提高了开发者的效率. 在本研究的敏捷软件开发中选择 Groovy. 其他影响因素还包括工具、文档、设计及人员的编程能力.

分布开发任务分为三个层次: 分布开发总任务层、子任务(子团队)层和子任务中的各个具体任务层. 如今软件开发将一个开发任务分给多个分布式子团队, 子团队实施具体开发步骤, 具体包括选择开发语言、设计、编写文档等. 由于各个子团队并行工作, 因此, 总任务完成的前提是各个子团队的任务必须全部完成. 根据以上分布开发任务的层次结构, 可以建立分布开发任务的信息模型.

2.2 任务分派算法

任务分派是将软件开发工作的各个具体环节和步骤按照一定的优化组合策略分解为由不同的分布开发子团队独立完成的子任务. 按照前述给出的任务分派模型, 异地分布开发任务由所有子团队的开发效率决定, 而分布式地区优势又会影响分布式子团队之间的优先分数, 每一个独立的团队在开发过程中有会有多方面的因素. 在任务分派过程中, 充分利用优势、避开劣势, 才会使效率达到最高. 因此, 分布式软件开发任务分派的核心过程是在限定的时间约束下, 对开发过程中的影响因子采用一定的分解策略进行组合优化的过程. 按优化目标的不同, 可有两种任务分派策略:

(1)在满足时间约束的前提下, 所用开发人员最少, 即以使用资源最少为原则.

(2)在满足时间要求的前提下, 整个软件开发活动所用时间最短. 投入所有可用开发人员, 包括本地和异地的, 并且充分发挥人员的优势, 以最快的速度完成软件开发任务.

在敏捷软件开发中, 综合考虑了软件开发中的地区、语言、交流、文化差异、人员水平等因素, 为尽可能提高开发效率, 提高开发质量, 故选择第二种分配策略, 同时, 利用任务分派算法, 合理分派开发任务, 尽可能提高开发效率, 也有利于第二个分派策略的实现. 如何充分利用地区优势及人员优势, 将整个开发活动时间缩到最短, 将在以下模型和算法中给出.

异地分布式软件开发, 是一个知识密集型的任务并贯穿于软件开发的各个阶段. 个人和团队需要及时有效的沟通、共享信息和协调. 为保证分布式开发顺利进行, 需要具备的基本要素: 一个有力的项目经理进行根据项目紧急程度、分布式地区优势进行任务分派. David K M 曾提出了计算多任务优先次序的传统的层次方法[19], 该方法较多的考虑到了理论上的影响因素, 并且仅给出了一次开发实践中的应用, 并没有对分派方法进行抽象以普遍适用于软件开发活动, 而软件开发注重实际应用, 故在此基础上进行了改进, 每个迭代周期均可采用该分派方法. 考虑到子团队分布式地区优势对最终运算结果影响较小, 并且项目决策者对地区优势已经非常了解, 因此简化了子团队分布式地区优势的算法, 同时充分考虑软件开发中的实际影响因素, 据此列出矩阵, 得出对异地分布式多个

子团队进行敏捷软件开发的优先分数计算方法. 根据实际开发经验, 本研究中的实际影响因素包括: 分布式地区优势(Regional Advantage)、各个团队的综合能力. 利用 Delphi 方法[20], 其观点数量多、观点质量高、财务成本低等, 作为一种主观、定性的方法, 不仅可以用于预测领域, 而且可以广泛应用于各种评价指标体系的建立和具体指标的确定过程. 具体步骤如下:

步骤 1: 建立评语集 RA_C = {C1, C2, ……, Cm};

步骤 2: 建立权重指标集: 利用 Delphi 方法, 使 m 个专家对各个分布式地区的权重赋值.

若干专家对 m 个子团队的分布式地区优势(RA)赋予权重, 评判结果用矩阵表示如下:

$$RA=\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \cdots \\ r_m \end{pmatrix}_m \tag{1}$$

影响 r 值的因素有: (1)分布式团队距离较远存在较大时差, 导致及时沟通、交流困难, 对团队协作造成最严重的困难; (2)信息流动不规律, 不能及时将需求变化或其他问题及时发布或反馈; (3)文化差异, 相互理解存在较大困难.

根据开发中语言、文档、设计、工具等工作关键程度的不同, 可得到表 1.

表 1 影响因素关键程度列表

	Groovy	Document	Design	Tools
Groovy	1			
Document		1		
Design			1	
Tools				1

表中的每一个元素表示列中表示的属性与行中表示的属性相比的重要程度, 两个指标之间重要性判断值如表 2 所示[21], 根据软件开发经验及人员水平进行取值.

表 2 指标重要性判断矩阵中元素的取值

相对重要程度	意义
1	同等重要
3	略微重要
5	相当重要
7	明显重要
9	绝对重要
2,4,6,8	两个相邻判断的中间值, 在需要折中时采用

由于在软件开发中的影响因素不同, 为获得研究的一般性, 将上表抽象为矩阵表示如下:

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}_{n\times n} \tag{2}$$

其中 a_{ij} 表示技能 j 与技能 i 相比对软件开发的影响程度, 如 a_{13}=5, 表示第三种技能与第一种技能相比相当重要, 且重要性为 5.

矩阵 A 为非零矩阵, 因每一种技能对软件开发都会产生较大影响, 故矩阵中的元素不能为零, 可以利用 MATLAB 进行操作. 显然, 矩阵 A 应满足一下约束条件:

(1)矩阵 A 为一个 n×n 方阵, 且方阵中任意一个元素不能为零, 如果两种技能对软件开发的影响相同, 则表示为 1; 对软件开发的影响越大则值越大.

(2)矩阵 A 中主对角线上的元素均为 1, 为同一种技能之间的比较.

(3)矩阵 A 可以看做是以主对角线为对称轴的特殊的对称矩阵, 因互为对称位置的两个元素互为倒数.

同理, 可以得到表示 n 个不同团队编程人员编程能力的表格 3, 进而抽象为矩阵, 如表 3 所示.

表 3 开发人员编程能力列表

	A	B	C	……	n
A	1				
B		1			
C			1		
……				……	
n					1

矩阵表示如下:

$$B=\begin{pmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1m} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2m} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & b_{m3} & \cdots & b_{mm} \end{pmatrix}_{m\times m} \tag{3}$$

同理可知, 矩阵 A 与矩阵 B 满足相同的约束条件.

算法步骤如下:

步骤 1: 将矩阵 A 归一化后得出行平均值, 记为列

向量 $\alpha=\begin{pmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \cdots\\ \alpha_n\end{pmatrix}_n$，定义为综合技能定额向量，表示对软件开发的综合影响度;

步骤 2: 同理，将矩阵 B 归一化后得出行平均值，记为列向量 $\beta=\begin{pmatrix}\beta_1\\ \beta_2\\ \beta_3\\ \cdots\\ \beta_m\end{pmatrix}_m$，定义为能力定额向量，为 m 个团队各编程人员的编程能力;

步骤 3: 将能力定额向量与综合技能向量的转置相乘，得出各个子团队的工作能力，记为 $C_{m\times n}=\beta\times\alpha^T$；得出行平均值，定义为综合能力定额向量，记为 γ.

$$\gamma=\begin{pmatrix}\gamma_1\\ \gamma_2\\ \gamma_3\\ \cdots\\ \gamma_n\end{pmatrix}_m$$

步骤 4: m 个分布式开发团队任务分派(TA)结果: $TA_m=\gamma\times RA_m^T$，取行平均值，归一化，得到矩阵 θ.

$$\theta=\begin{pmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \cdots\\ \theta_m\end{pmatrix}_m$$

可将全部任务看做单位 1，根据 θ 值即可将任务分派给 m 个异地分布式团队;

步骤 5: 置迭代次数 k，引入循环变量 i;

for(i=1;i<=k;i++)

{

步骤 6: 根据上次迭代开发结果与开发计划比较，如需调整，则执行步骤 7；无需调整，重复步骤 1-4;

步骤 7: 调整影响因素的关键程度，一般情况下为提高指标重要性判断矩阵中元素的取值，使之匹配开发计划;

步骤 8: 在每次迭代中重复步骤 6-8，直到项目结束.

}

Matlab 提供了一个简单的归一化函数[x,ps] = mapminmax(xmin,xmax,y),计算可得到一列归一化后的数据，然后利用矩阵乘法得出分派结果.

3 计算实例

3.1 算例描述

以 H 公司软件开发经验为例；该公司总部设于美国，其他两个团队分别分布于中国和印度. 由于篇幅原因，省去利用 Delphi 方法进行专家调查的具体过程，综合考虑通信、语言、文化等差异，并参考以往开发经验，得出地区优势分别为 1、0.6、0.4，给出分布式地区优势矩阵 RA；对于每一个分布式团队，分别考虑每个团队使用的开发语言、所选工具、工程设计、文档的编写等因素，得出软件开发影响因子矩阵 A，应符合上文给出的矩阵约束条件；同理，结合每个团队成员的编程能力，得到能力矩阵 B. 按照算法步骤，仅给出一次算法结果.

3.2 任务分派的具体实施

根据 3.1 中的算例描述，给出异地开发团队地区优势矩阵 RA=$\begin{pmatrix}1\\ 0.6\\ 0.4\end{pmatrix}$;

A=$\begin{pmatrix}1 & 2 & 3 & 5\\ 0.5 & 1 & 2 & 4\\ 0.33 & 0.2 & 1 & 2\\ 0.2 & 0.25 & 0.5 & 1\end{pmatrix}$;

B=$\begin{pmatrix}1 & 0.33 & 0.2\\ 3 & 1 & 0.33\\ 5 & 3 & 1\end{pmatrix}$

步骤 1: 归一化矩阵 A，得出 $\alpha=\begin{pmatrix}0.49\\ 0.30\\ 0.14\\ 0.08\end{pmatrix}$;

步骤 2: 归一化矩阵 B, 得出 $\beta=\begin{pmatrix}0.11\\0.26\\0.33\end{pmatrix}$;

步骤 3:

$$C=\beta\times\alpha^{T}=\begin{pmatrix}0.054 & 0.033 & 0.015 & 0.009\\0.127 & 0.078 & 0.036 & 0.021\\0.309 & 0.189 & 0.088 & 0.050\end{pmatrix},$$

$$\gamma=\begin{pmatrix}0.028\\0.066\\0.159\end{pmatrix};$$

步骤 4:

$$TA_m=\gamma\times RA_m{}^{T}=\begin{pmatrix}0.028 & 0.017 & 0.011\\0.066 & 0.040 & 0.026\\0.159 & 0.095 & 0.064\end{pmatrix},$$

$\theta=\begin{pmatrix}0.11\\0.24\\0.65\end{pmatrix}$. 若将所以工作量看做单位 1, 则美国总部、中国、印度三个团队任务分派分别为 0.11、0.24、0.65.

该算法适合异地分布式敏捷软件开发. 首先, 算法中的各个影响因素非常清晰, 项目成员可以自己检查各个影响因素. 在一次或几次迭代周期后, 如果未按照计划完成任务, 项目经理可以检查各个影响因素重新调节各个权值, 使其按照项目计划进展. 如果客户要求提前项目交付日期, 项目经理也可以依照以上分派方法重新分派任务. 其次, 任务分派完成之后, 在保证团队整体工作能够按时完成情况下, 各个子团队成员可以根据自身情况选择适合自己的任务.

4 结语

任务分派应用于多个团队, 尤其是异地分布式团队. 由于处于分布的各个团队存在地域上的差别, 不能像一般的敏捷开发团队进行面对面的沟通和交流、站立会议等, 决定了异地分布式团队的任务分派要区别于其他软件开发的简单任务分派. 本文提出的异地分布式任务分派综合考虑了地域、分布式环境下人员的各项能力等因素, 将各种因素基于一定的规则列于矩阵中, 进行运算后得出任务分派的结果. 此外, 该算法适合于任意多个分布式开发团队, 并且分布式团队越多, 利用该方法进行分派在理论上越高效. 在任务分派过程中难免会存在偏差, 可以在任务执行过程中对任务进行监控, 与计划进度进行比较, 尽量减小偏差, 以保证软件开发工作如期完成.

参考文献

1 Damian D, Moitra D. Global software development: How far have we come? IEEE Softwear, 2006, 23(5):17–19.

2 Ansgar L, Jürgen M, Dieter R. Towards a multi-criteria development distribution model: An analysis of existing task distribution approaches. International Conference Global Software Engineering. US. IEEE. 2008. 109–118.

3 Chua AL, Pan S. Knowledge transfer in offshore insourcing. Proc. of the 27th International Conference on Information Systems. US. Association for Information Systems. 2006. 1039–1053.

4 Sakthivel S. Managing risks in offshore systems development . Commun. ACM. 2007, 50(4): 69–75.

5 Hei CM, Herbsleb JD. End-to-end features as meta-entities for enabling coordination in geographically distributed software development. Software Development Governance. US. IEEE Computer Society. 2009. 21–26.

6 Narayan R, Rajesh KB. Globally distributed software development project performance: An empirical analysis. Proc. of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. US. ACM. 2007.125–134.

7 Michael AC. Managing software development in globally distributed teams. Communication of the ACM-Alternate Reality Gaming, 2008, 51(2): 15–17.

8 李春芳,谭庆平,乐晓波,肖晓丽.基于多机制与策略约束的分配元模型.计算机技术与发展,2009,19(2):1–8.

9 何炎祥,罗先林,吴思,彭堂玉,祝向庆.对三种典型分布式任务分配算法的分析.小型微型计算机系统,1997,18(11):1–6.

10 郭希娟,李墨华.基于多准则的任务分配算法.计算机应用,2008,28(10):2507–2512.

11 张洁,张朋,刘国宝.基于两阶段蚁群算法的带非等效并行机的作业车间调度.机械工程学报,2013,49(6).

12 Ansgar L. Model-based task allocation in distributed software

development. Software Engineering Approaches for Offshore and Outsourced Development-4th International Conference. Germany. Springer Verlag. 2010. 37–53.

13 Ansgar L, Jürgen M. A multi-criteria distribution model for global software development projects. Journal of the Brazilian Computer Society, 2010, 16(2): 97–115.

14 Marques AB, Carvalho JR, Rodrigues R, Conte T, Prikladnicki R, Marczak S. An ontology for task allocation teams in distributed software development. International Conference on Global Software Engineering. US. IEEE Computer Society. 2013. 21–30.

15 Shim J, Lee S, Wu C. A unified approach for software policy modeling: Incorporating implementation into a modeling methodology. Lecture Notes in Computer Science, 2003, 28(13): 118–130.

16 Shen M, Tzeng GH, Liu DR. Multi-criteria task assignment in workflow management systems. Proc. of the 36th Annual Hawaii International Conference on Social Science, 2003.

17 Duggan J, Byrne J, Lyons GJ. A task allocation optimizer for software construction. IEEE Software, 2004, 21: 76–82.

18 Sengupta B, Chandra S, Sinha V. A research agenda for distributed software development. Proc. of the 28th International Conference on Software engineering. US. IEEE. 2006. 731–740.

19 David KM, Philippe BK. Task coordination in an agile distributed software development environment. Canadian Conference on Electrical and Computer Engineering. US. IEEE. 2007. 606–611.

20 幸莉仙,韩玢,黄慧莲.AHP 与 FUZZY 在软件开发项目风险管理中的综合应用.计算机系统应用,2012,21(4):161–164.

21 张纯.基于模糊层次分析法的构件评估.湖南第一师范学报,2010,10(3):154–156.