计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(4):236−241 [doi: 10.15888/j.cnki.csa.007365]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于能力匹配的异地敏捷开发任务分配方法①

叶上华[1,2], 殷　茗[2], 杨　益[2], 姜继娇[3]

[1](西北工业大学 计算机学院, 西安 710072)
[2](西北工业大学 软件学院, 西安 710072)
[3](西北工业大学 管理学院, 西安 710072)

摘　要: 为了能够快速准确地获得异地敏捷软件开发团队任务分配的全局最优解, 提出了一种基于能力匹配的异地敏捷开发任务分配方法. 该方法强调子任务能力需求和团队能力的匹配关系, 构建了能力匹配的效用函数, 对效用矩阵进行求解, 全局效用值最大时获得最优分配方案. 算例仿真结果表明, 所提出的方法可以有效得到能力匹配较优的任务分配方案.

关键词: 异地敏捷开发; 任务分配; 效用; 匈牙利算法

引用格式: 叶上华,殷茗,杨益,姜继娇.基于能力匹配的异地敏捷开发任务分配方法.计算机系统应用,2020,29(4):236–241. http://www.c-s-a.org.cn/1003-3254/7365.html

Distributed Agile Development Task Allocation Based on Capability Matching

YE Shang-Hua[1,2], YIN Ming[2], YANG Yi[2], JIANG Ji-Jiao[3]

[1](Institute of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)
[2](Institute of Software, Northwestern Polytechnical University, Xi'an 710072, China)
[3](Management School, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: In order to quickly and accurately obtain the global optimal solution for task allocation of distributed agile software development teams, the study proposes a method based on capability matching. This method builds the utility function of capability matching on the basis of the subtask capability demand degree vector and the team ability vector. We solve the utility matrix, and the optimal allocation scheme is obtained when the global utility value is the largest. The simulation results show that the method could effectively obtain a task allocation scheme with better capability matching.

Key words: distributed agile development; task allocation; utility; Hungarian algorithm

异地敏捷开发已成为软件企业开发项目 IT 策略的一部分[1], 这种开发可能在一个单独项目中涉及到多个地点, 或软件开发周期中一部分的服务提供者[2]. 敏捷方法已存在多年, 通过使用迭代和增量方法为业务增加价值, 其中软件开发人员处理小模块并响应用户不断变化的业务需求, 并在价值、原则和实践方面为软件工程带来了重大变化. 因此, 敏捷方法通过接受“反馈和变化”以及“拥抱而不是拒绝更高的变化率”来提供灵活性以及对软件开发过程的严谨性[3]. 近年来, 敏捷方法在全球化环境中越来越受到重视, 其低成本, 早期产品交付和高质量产品等多种优势使得异地敏捷开发成为当下诸多企业积极探索的趋势. 然而, 一些研究指出, 由于异地敏捷开发存在文化差异、时间差异、地理分布等诸多特征[4], 在全球分布式环境中采用

① 基金项目: 航空科学基金 (2016ZG53071); 教育部人文与社会科学基金 (16YJA630068, 18YJA630043); 陕西省自然科学基础研究计划 (2018JM7008); 陕西省社会科学基金 (2018S28)
Foundation item: Aviation Science Fund (2016ZG53071); Humanity and Social Science Fund of Ministry of Education, China (16YJA630068, 18YJA630043); Fundamental Research Program of Natural Science of Shaanxi Province (2018JM7008); Social Science Fund of Shaanxi Province (2018S28)
收稿时间: 2019-08-28; 修改时间: 2019-09-29; 采用时间: 2019-10-24; csa 在线出版时间: 2020-04-05

敏捷方法会带来重大挑战, 尤其是一些敏捷原则和实践强调近距离、面对面交互. 敏捷开发实践需要频繁地非正式通信, 例如每日站立会议, 以便管理项目并实现 Scrum 的效果, 当成员在地理上分散时, 知识共享难以实现、非正式沟通很难进行[5]. 任务分配作为软件项目的基础活动, 是异地分布式敏捷软件开发中的关键环节[1], 分配决策需要考虑多方面因素, 任务分配可减少异地敏捷团队之间的沟通成本, 使团队成员能把大部分精力投入到软件开发[6].

目前, 研究者们对任务分配的研究大多主要集中在人工智能领域, 如多 Agent 系统[7]、机器人[8]等. 任务分配的目的, 是为了合理配置现有资源, 不同应用需要不同的任务分配机制. 现有研究[8,9]大多致力于集中式任务分配, 未考虑异地敏捷团队的分布式的能力和资源. 例如 Butler 和 Heys[10]提出了基于行为和拍卖的任务分配方法, 行为方法提出了高中低的自重构团队行为及六个任务行为, 采用动机函数平衡不同行为完成任务布置, 拍卖方法解决任务分配中的行为交互, 及添加额外交互行为. Pendharkar[11]提出了启发式蚁群的任务分配优化方法, 分析了约束任务分配的线性整数规划及其松弛规划、低限制拉格朗日松弛, 提出了约束任务分配的迭代启发式贪婪及基于启发式蚁群算法的约束任务分配模型. Wilson 和 Quigley[12]提出了可靠性增长的任务分配模型, 从项目风险视角出发, 以成本和时间为约束, 识别和设计系统开发可能的任务弱点, 从成本、时间测试系统可靠性, 建立多属性效用函数, 构建可靠性增长任务分配模型. 上述研究虽然分别解决了任务布置和行为交互、约束任务分配及风险视角任务分配问题, 但普遍忽略了任务与异地团队的多属性能力匹配.

异地敏捷软件开发中的任务要分配给异地团队, 这些异地团队具有不同的能力和资源来完成这些任务, 因此需要考虑异地敏捷环境的多个决策属性. 现有学者已经对此展开了初步研究. Duggan 等[13]认为软件构建中的任务分配需要考虑多个目标, 包括成本、缺陷数量、完成时间、员工使用与客户满意度等, 采用演化算法和遗传算法进行多目标任务分配可以达到优化效果. Ruano-Mayoral 等[14]提出了一个全球开发项目任务包二阶段分配框架, 提出了任务分配的决策影响因子, 包括可用性和能力; Almeida 等[15]提出了一个异地分布式开发认知映射和 MACBETH 的多维决策模型, 其中决策因子包括成本、分布水平、文化差异水平、团队年龄、团队技能适应性、沟通有效性、沟通频率、协作控制水平等. Lamersdorf 和 Münch[16]提出了全球软件开发的一个客户化多维需求任务分配模型, 运用了改进 Bokhari 算法, 该模型在多个典型假设场景和实际分布决策问题获得了应用效果. 张立等[17]提出半自治多 Agent 任务分配方法, 将心智模型与扩展合同网机制结合, 并扩展了合同网机制包括发标优选、竞标报价与多 Agent 任务分配过程. 另外, 殷茗和马静[18]提出了基于多任务优先算法的异地软件开发任务分配模式, 该研究主要关注异地分布式软件分配的优先权属性.

上述研究为本研究提供了很好的研究借鉴, 但是没有考虑任务与团队的多属性能力匹配, 对异地敏捷团队的任务分配效用没有聚焦. 本文充分考虑了异地敏捷开发的特征和挑战, 首先提炼出该环境下的子任务能力需求属性和团队能力属性, 包括了技术能力、协调能力、创新能力和敏捷开发能力; 其次, 在整数线性规划的基础上提出了基于能力匹配的效用函数, 分别求解出了任务和团队的能力匹配效用值; 再次, 以全局效用最大化为目标, 采用匈牙利算法求解出了最优的任务分配方案; 最后, 通过算例仿真验证了方法的有效性, 从而为异地敏捷软件开发任务分配的决策实践提供理论依据.

1 基于能力匹配的异地敏捷开发任务分配方法

1.1 问题描述

假设异地敏捷开发中待分配的任务已经分解为 n 个子任务 $T_i(i=1,2,3,\cdots,n)$, 参与项目开发的 n 个团队 $P_j(j=1,2,3,\cdots,n)$, 由多个异地团队通过敏捷开发方式协作完成目标任务 T. 不同的子任务 T_i 对团队 P_j 有不同的能力要求, 而团队 P_j 有自己的能力, 团队选择不同任务产生的效用不同. 效用函数就是用来评价任务和团队的匹配程度的函数. 因此, 本文在根据任务能力需求属性和团队能力属性的基础上, 利用效用函数将任务分配给最合适的团队执行, 从而实现全局最优.

异地分布式无法支持敏捷开发所提倡的面对面的沟通与交流, 由此出现了大量待解决的协作问题. 协作是“管理活动之间的依赖关系”, 这种依赖关系包括共享的资源、任务分配、任务和子任务之间的关系[19], 团队协作、目标导向、团队内聚力、共享心智模式及团队学习等是影响软件开发的重要变量[20]. 敏捷方法

是高度协作的, 无论是在开发组内部还是在开发组之间. 敏捷方法依靠非正式沟通而不是大量文档来快速传播整个团队和其他利益相关者的信息. 没有高度协作的环境, 任何敏捷方法都注定要失败. 由于敏捷方法在很大程度上依赖于协作和沟通[21], 因此团队是成功的关键. 考虑异地敏捷开发协作特征, 本文将协作能力作为任务和团队的重要能力属性. 另一方面, 异地团队开发的“敏捷性”, 可以快速适应不稳定的市场环境和需求波动[22], 即适应人员配置、程序设计、系统体系架构、软件开发流程和软件如何遵照业务调整和预算结构的变更[23]. 因此, 将敏捷开发能力作为异地敏捷开发任务分配的能力属性. 对于技术能力和创新能力的提出则是因为敏捷方法的最大影响在于开发团队. 敏捷方法依赖于强大的开发团队, 除了具备较强的沟通协作能力和敏捷开发的经验, 开发团队还必须技术娴熟、思维活跃、善于变通, 这不仅仅异地敏捷开发团队更是任何软件开发团队应该具备的优秀品质. 根据上述分析, 结合软件开发团队强调技术与创新能力的特征, 本研究认为异地敏捷开发的子任务T_i有 4 个能力属性: 协作能力、敏捷开发能力、技术能力和创新能力. 采用能力需求向量表示各个能力的需求值, 如式 (1) 所示.

$$T_i\left[x_i^1,x_i^2,\cdots,x_i^k\right](k=1,2,3,4) \tag{1}$$

与子任务能力属性相对应, 异地敏捷团队P_j拥有 4 个能力属性: 协作能力、敏捷开发能力、技术能力和创新能力. 用能力拥有向量表示各个能力的拥有值, 如式 (2) 所示.

$$P_j\left[y_j^1,y_j^2,\cdots,y_j^k\right](k=1,2,3,4) \tag{2}$$

1.2 基于能力匹配的效用函数

要实现整个系统优化, 应当采用明确的标度度量任务分配结果, 由此有必要分析基于能力匹配的效用函数. 随着环境变化, 任务需求能力和团队能力都随之变化, 团队对子任务的胜任程度也相应变化[24], 因此, 基于能力匹配的效用需要重点考虑团队能力对任务所需能力的满足程度. 任务和团队的能力匹配研究不仅可以最大程度地调动开发团队的工作潜能和主观能动性, 还可以使项目完成实现效用最大化[25], 从而使软件开发呈现良性循环的模式. 效用函数U_{ij}是一个量化第 j 个团队和第 i 个子任务匹配度的函数.

本文主要研究异地敏捷开发的任务分配中任务能力需求度和团队拥有能力的匹配程度, 分别包括技术能力、协调能力、创新能力和敏捷开发能力. 与普遍情况下的软件开发任务分配不同, 本文的效用函数充分结合异地敏捷开发的特点对能力类型进行严格筛选, 考虑到异地敏捷开发技术较为先进、异地沟通成本大、采用敏捷开发的软件较为新颖和敏捷开发团队规模较小等因素, 最终效用函数确定技术能力、协调能力、创新能力和敏捷开发能力这 4 种能力为本文研究的主要能力, 从而使得效用函数的使用更加符合异地敏捷开发的任务分配研究背景. U_{ij}计算公式如式 (3) 所示.

$$U_{ij}=\sum_{k=1}^{k=4}\omega_{ij}^k\times\partial_{ij}^k \tag{3}$$

$$\partial_{ij}^k=1-\frac{\left|y_j^k-x_i^k\right|}{Y_j-X_i},x_i^k,y_j^k\in\left[X_i,Y_j\right] \tag{4}$$

$$\omega_{ij}^k=\frac{\text{任务}i\text{第}k\text{个能力的需求值}}{\text{任务}i\text{所有能力的总需求值}} \tag{5}$$

其中, x_i^k为任务 i 的对于第 k 个能力的需求度, y_j^k为团队 j 对于第 k 个能力的实际拥有值, $x_i^k\in[a_1,b_1]$, $y_j^k\in[a_2,b_2]$, $X_i=\min(a_1,a_2)$, $Y_j=\max(b_1,b_2)$. ∂_{ij}^k为任务 i 和团队 j 对于第 k 个能力的单一能力匹配程度. 由于团队的第 k 个能力可能大于任务的需求值也可能小于任务的需求值, 而这两种情况对单一能力匹配度的影响程度相同, 即当团队的某种能力高于任务需求某种能力或者团队的某种能力低于任务需求某种能力时会相同程度的降低匹配度, 只有团队的某种能力与需求能力越相近, 单一能力匹配度才会越高, 故而添加绝对值符号. ω_{ij}^k为针对任务 i 和团队 j, 第 k 个能力的权重. 最终得效用函数矩阵U_{ij}如式 (6) 所示. 基于异地敏捷开发的任务分配就是在效用函数矩阵U_{ij}中, 选择 n 个不同行不同列的元素求解, 根据算法选择效用值最大的方案即为最优分配方案.

$$U=\begin{bmatrix}u_{11}&u_{12}&\cdots&u_{1n}\\u_{21}&u_{22}&\cdots&u_{2n}\\\vdots&\vdots&\ddots&\vdots\\u_{n1}&u_{n2}&\cdots&u_{nn}\end{bmatrix} \tag{6}$$

1.3 效用矩阵求解

针对 1.2 节所提出的基于异地敏捷开发的任务分配的效用函数所构成的效用矩阵求解, 本文采用匈牙

利算法求解效用矩阵问题. 匈牙利算法作为求解指派问题的经典算法, 在避免程序陷入无限循环的情况下, 可以求得指派问题精确的全局最优解. 无论是对普通的软件任务分配问题还是对结合异地敏捷开发特色的任务分配, 该算法都有很好的适应性. 效用计算步骤如下.

Step 1. 从效用矩阵的每行最大匹配值减去该行元素, 再从效用矩阵的每列最大匹配值减去该列元素.

Step 2. 此时效用矩阵每行每列都存在零元素, 需找出 n 个独立的零. 若能找出, 就以这些独立零对应的效用矩阵中的匹配度为 1, 其余为 0, 得到最大效用值. 当 n 较小时, 可用观察法、试探法找出 n 个独立零元素. 若 n 较大, 则需要按一定顺序去找, 具体如下:

(1) 从只有一个零的行 (列) 开始, 给这个零加圈, 划去画圈的列 (行) 中的其他零;

(2) 给只有一个零列 (行) 的零加圈, 然后划去加圈的元素所在行 (列) 的零;

(3) 反复进行 Step 2 中 (1)、(2) 两步, 直到所有零都被圈出或划掉为止;

(4) 若仍有没有画圈的零, 且同行 (列) 的零至少有两个. 从剩有零最少的行 (列) 开始, 比较这行零所在列中零的数目, 选择列中零最少的零加圈, 然后划掉同行同列的其他零. 反复进行, 直到所有零都圈出或划掉为止;

(5) 若画圈零元素的数目 m 等于矩阵的维数 n, 那么已得到效用值最大的分配结果; 若 $m<n$, 则转下一步.

Step 3. 做最少的直线覆盖所有零, 以确定该代价矩阵中能找到最多的独立零.

(1) 对没有画圈的行打对号;

(2) 对已经打对号的行中有划掉零的列打对号;

(3) 再对打有对号的列中含画圈元素的行打对号;

(4) 重复 Step 3 中 (2)、(3), 直到得不出新的打对号的行、列为止;

(5) 对没有打对号的行画横线, 对打对号的列画纵线, 得到覆盖所有零的最少直线数.

Step 4. 经过上述变换得到新的效用矩阵. 在没有被直线覆盖的数中找出最小值, 并对没划直线行的各数都减去该最小值, 对划直线列的各数都加上该最小值, 得到新矩阵, 转 Step 2.

2 算例仿真

本文的核心是研究异地敏捷开发环境条件下合理的任务分配过程, 研究对象是敏捷开发任务和异地敏捷开发团队. 传统人力处理任务分配, 无法保证分配结果质量. 本文提出的基于能力匹配的方法可以量化分配结果, 保证全局最优解. 为验证该方法的有效性, 本研究在异地敏捷开发某购物网站项目为背景, 利用效用函数进行任务分配案例仿真. 仿真环境采用 17.12 版本 CodeBlocks, C 语言编程, 在 CPU 2 GHz、 4 GB 内存、256 GB 硬盘空间的 64 位 Windows 8 主机运行. 前期采用问卷调查获取仿真初始数据.

2.1 实验仿真

数据采集通过问卷调查获取, 以 Likert5 级量表对异地敏捷开发任务能力需求度属性和异地敏捷开发团队能力属性进行打分. 现将任务名称用任务标号来表示, 如表 1 所示, 并做出如下规定: 异地敏捷开发任务对技术能力, 协调能力, 创新能力和敏捷开发能力的需求值用 (0,1], (1,2], (2, 3], (3, 4], (4, 5]区间的右端点{1,2,3,4,5}表示, 分值越高表示需要程度越高; 同理, 异地敏捷开发团队的技术能力, 协调能力, 创新能力和敏捷开发能力拥有值也用 (0, 1], (1, 2], (2, 3], (3, 4], (4, 5]区间的右端点{1,2,3,4,5}表示. 最终数据如表 2 和表 3 所示.

表 1 任务标号和任务名称的对应关系

任务标号	任务名称
任务 1	用户注册登录购物网站模块
任务 2	用户在网站上浏览商品并购买模块
任务 3	用户在网站上浏览商品并收藏模块
任务 4	用户对商品的评论和询问模块
任务 5	管理员管理商品模块
任务 6	管理员管理用户模块
任务 7	管理员处理订单模块
任务 8	管理员管理评论和回复模块

表 2 异地敏捷开发任务能力需求数据

任务标号	技术能力	协调能力	创新能力	敏捷能力
任务 1	1	1	2	2
任务 2	3	2	3	3
任务 3	3	4	4	4
任务 4	2	4	3	4
任务 5	3	3	4	4
任务 6	2	2	4	4
任务 7	3	2	2	3
任务 8	1	3	2	3

表 3 异地敏捷开发团队能力数据

团队标号	技术能力	协调能力	创新能力	敏捷能力
团队 1	3	4	5	5
团队 2	1	4	2	3
团队 3	4	2	3	3
团队 4	3	1	5	4
团队 5	2	3	4	5
团队 6	5	4	1	4
团队 7	3	5	4	4
团队 8	4	5	4	5

按式 (4) 进行效用值计算, 首先求出异地敏捷团队和异地敏捷开发任务能力匹配时的效用值, 再采用匈牙利算法对效用矩阵进行求解. 采用 C 语言编程, 对实验结果用文本输出的方式表示. 效用矩阵结果表 4 所示.

本文为数据处理方便, 将效用矩阵乘以 1000 后再按照匈牙利算法求最大值进行处理; 式 (7) 是最终分配矩阵, 1 表示分配, 0 表示不分配.

表 4 任务和团队组的效用矩阵结果

任务标号	团队 1	团队 2	团队 3	团队 4	团队 5	团队 6	团队 7	团队 8
任务 1	0.264	0.708	0.458	0.417	0.333	0.353	0.311	0.269
任务 2	0.515	0.561	0.864	0.591	0.455	0.379	0.591	0.409
任务 3	0.733	0.556	0.456	0.667	0.633	0.667	0.867	0.633
任务 4	0.615	0.654	0.538	0.538	0.577	0.731	0.654	0.474
任务 5	0.607	0.417	0.500	0.714	0.750	0.536	0.857	0.607
任务 6	0.472	0.471	0.556	0.667	0.750	0.514	0.792	0.597
任务 7	0.517	0.667	0.750	0.600	0.417	0.417	0.567	0.367
任务 8	0.370	0.833	0.639	0.370	0.574	0.467	0.389	0.324

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

经过 23 次迭代, 式 (7) 为整体效用最高时的分配矩阵, 最终分配方案为: 任务 1 和团队 2、任务 2 和团队 3、任务 3 和团队 1、任务 4 和团队 6、任务 5 和团队 7、任务 6 和团队 8、任务 7 和团队 4、任务 8 和团队 5.

2.2 结果分析

异地敏捷开发项目匹配效用量化值如表 5 所示, 且效用值为 4.837. 效用值大小是由任务能力需求、团队能力及任务和团队的匹配程度来决定. 计算后可以得出每一个任务和团队的匹配度的衡量标准都在 0 到 1 区间里取值, 故而整体效用值理论上是在 0 到 8 之间. 由此可得该算例的分配结果最高效用适中.

表 5 匹配效用量化值

匹配方式	(任务 1, 团队 2)	(任务 2, 团队 3)	(任务 3, 团队 1)	(任务 4, 团队 6)	(任务 5, 团队 7)	(任务 6, 团队 8)	(任务 7, 团队 4)	(任务 8, 团队 5)
U_{ij}	0.515	0.380	0.242	0.550	0.867	0.769	0.750	0.607

当前异地敏捷团队的能力值均高于任务能力需求值, 可见并非能力越高的团队产生的效用越高, 对于高能力团队接受低能力需求的任务时, 同样是人才资源的浪费. 只有任务和团队二者的能力刚好彼此满足时, 才能产生较高的效用值. 这为日后任务分解的改进或参与异地敏捷项目团队的调整给出理论性的依据.

效用值作为直接定量描述异地敏捷任务和异地敏捷团队分配方案质量的数值, 其越大说明任务和团队能力越匹配, 分配方案越合理. 实际运用中, 管理者应当在任务发布初期遵循效用值高的异地敏捷团队参与任务的原则进行团队选择, 只有这样才会更高质量的完成任务.

3 结论

本文充分考虑了软件异地敏捷开发的任务分配特点, 提炼出了异地敏捷开发环境下的任务能力需求属性和团队能力属性, 创新性地提出基于能力匹配的效用函数, 将分配结果量化表达, 使任务分配给最适合的团队执行, 达到了结果最优, 从而产生效用最高的分配方案. 通过算例验证本文任务分配方法的有效性. 但由于效用函数是基于线性规划思想, 对大规模的任务分配还缺乏普适性. 异地敏捷开发在软件开发领域已然兴起, 还有许多工作需要将来继续深入研究.

参考文献

1 Sutanto J, Kankanhalli A, Tan BCY. Investigating task coordination in globally dispersed teams: A structural contingency perspective. ACM Transactions on Management Information Systems, 2015, 6(2): Article No.5.

2 Ågerfalk PJ, Fitzgerald B, Slaughter SA. Introduction to the Special Issue—Flexible and distributed information systems development: State of the art and research challenges. Information Systems Research, 2009, 20(3): 317–328. [doi: 10.1287/isre.1090.0244]

3 Modi S, Abbott P. Understanding collaborative practices in distributed agile development: Research proposal. Proceedings of 2013 IEEE 8th International Conference on Global Software Engineering Workshops. Bari, Italy. 2013. 74–77.

4 Shameem M, Kumar RR, Kumar C, *et al*. Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process. Journal of Software: Evolution and Process, 2018, 30(11): e1979. [doi: 10.1002/smr.1979]

5 Mak DKM, Kruchten PB. Task coordination in an agile distributed software development environment. Proceedings of 2006 Canadian Conference on Electrical and Computer Engineering. Ottawa, ON, Canada. 2006. 606–611.

6 Adelman L, Miller SL, Yeo C. Testing the effectiveness of icons for supporting distributed team decision making under time pressure. IEEE Transactions on Systems, Man, and Cybernetics —Part A: Systems and Humans, 2004, 34(2): 179–189. [doi: 10.1109/TSMCA.2003.819492]

7 Kim Y, Matson ET. A realistic decision making for task allocation in heterogeneous multi-agent systems. Procedia Computer Science, 2016, 94: 386–391. [doi: 10.1016/j.procs.2016.08.059]

8 Caraballo LE, Díaz-Báñez JM, Maza I, *et al*. The block-information-sharing strategy for task allocation: A case study for structure assembly with aerial robots. European Journal of Operational Research, 2017, 260(2): 725–738. [doi: 10.1016/j.ejor.2016.12.049]

9 Zhang K, Collins EG Jr, Shi DQ. Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. ACM Transactions on Autonomous and Adaptive Systems, 2012, 7(2): Article No.21.

10 Butler Z, Hays J. Task allocation for reconfigurable teams. Robotics and Autonomous Systems, 2015, 68: 59–71. [doi: 10.1016/j.robot.2015.02.001]

11 Pendharkar PC. An ant colony optimization heuristic for constrained task allocation problem. Journal of Computational Science, 2015, 7: 37–47. [doi: 10.1016/j.jocs.2015.01.001]

12 Wilson K, Quigley J. Allocation of tasks for reliability growth using multi-attribute utility. European Journal of Operational Research, 2016, 255(1): 259–271. [doi: 10.1016/j.ejor.2016.05.014]

13 Duggan J, Byrne J, Lyons GJ. A task allocation optimizer for software construction. IEEE Software, 2004, 21(3): 76–82. [doi: 10.1109/MS.2004.1293077]

14 Ruano-Mayoral M, Casado-Lumbreras C, Garbarino-Alberti H, *et al*. Methodological framework for the allocation of work packages in global software development. Journal of Software: Evolution and Process, 2014, 26(5): 476–487. [doi: 10.1002/smr.1618]

15 Almeida LH, Albuquerque AB, Pinheiro PR. A multi-criteria model for planning and Fine-Tuning distributed scrum projects. Proceedings of the 2011 IEEE Sixth International Conference on Global Software Engineering. Helsinki, Finland. 2011. 75–83.

16 Lamersdorf A, Münch J. A multi-criteria distribution model for global software development projects. Journal of the Brazilian Computer Society, 2010, 16(2): 97–115. [doi: 10.1007/s13173-010-0010-6]

17 张立, 王茜竹, 赵春江, 等. 基于心智与扩展合同网的半自治多智能体任务分配. 计算机集成制造系统, 2015, 21(11): 2885–2892.

18 殷茗, 马静. 基于多任务优先算法的异地敏捷软件开发任务分派. 计算机系统应用, 2015, 24(1): 128–134. [doi: 10.3969/j.issn.1003-3254.2015.01.023]

19 Malone TW, Crowston K. The interdisciplinary study of coordination. ACM Computing Surveys, 1994, 26(1): 87–119. [doi: 10.1145/174666.174668]

20 Dingsøyr T, Fægri TE, Dybå T, *et al*. Team performance in software development: Research results versus agile principles. IEEE Software, 2016, 33(4): 106–110. [doi: 10.1109/MS.2016.100]

21 Martakis A, Daneva M. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners. Proceedings of IEEE 7th International Conference on Research Challenges in Information Science. Paris, France. 2013. 1–11.

22 Curcio K, Navarro T, Malucelli A, *et al*. Requirements engineering: A systematic mapping study in agile software development. Journal of Systems and Software, 2018, 139: 32–50. [doi: 10.1016/j.jss.2018.01.036]

23 Vallon R, da Silva Estácio BJ, Prikladnicki R, *et al*. Systematic literature review on agile practices in global software development. Information and Software Technology, 2018, 96: 161–180. [doi: 10.1016/j.infsof.2017.12.004]

24 Licorish SA, MacDonell SG. Exploring the links between software development task type, team attitudes and task completion performance: Insights from the Jazz repository. Information and Software Technology, 2018, 97: 10–25. [doi: 10.1016/j.infsof.2017.12.005]

25 Ceri-Booms M, Curşeu PL, Oerlemans LAG. Task and person-focused leadership behaviors and team performance: A meta-analysis. Human Resource Management Review, 2017, 27(1): 178–192. [doi: 10.1016/j.hrmr.2016.09.010]