·体系结构与软件技术· 文章编号：1000-3428(2019)07-0066-05 文献标志码：A 中图分类号：TP311.52

基于核心度排序的软件众包模块分配算法

王晨旭[1]，余敦辉[1,2]，张万山[1,2]，张兴盛[1]

（1. 湖北大学 计算机与信息工程学院，武汉 430062；2. 湖北省教育信息化工程技术研究中心，武汉 430062）

摘 要： 为提升众包任务分配效率，提出一种改进的软件众包模块分配算法。根据结构复杂度与技术复杂度获得待开发模块复杂度，采用余弦相似度算法估测模块质量，依据关键路径得出模块重要度，综合模块复杂度、质量及重要度计算模块核心度，通过模块核心度排序实现软件众包模块分配。实验结果表明，与复杂度优先、质量优先、重要度优先的分配算法相比，该算法的适配值至少可提升 33.9、27.7 及 27.8。

关键词： 软件众包；模块分配；余弦相似度；模块核心度；适配值

开放科学（资源服务）标志码（OSID）：

中文引用格式： 王晨旭，余敦辉，张万山，等. 基于核心度排序的软件众包模块分配算法[J]. 计算机工程，2019，45(7)：66-70.

英文引用格式： WANG Chenxu，YU Dunhui，ZHANG Wanshan，et al. Module allocation algorithm for software crowdsourcing based on core degree sorting[J]. Computer Engineering，2019，45(7)：66-70.

Module Allocation Algorithm for Software Crowdsourcing Based on Core Degree Sorting

WANG Chenxu[1]，YU Dunhui[1,2]，ZHANG Wanshan[1,2]，ZHANG Xingsheng[1]

（1. School of Computer Science and Information Engineering，Hubei University，Wuhan 430062，China；
2. Hubei Province Engineering Technology Research Center for Education Informationization，Wuhan 430062，China）

【Abstract】 To increase the efficiency of crowdsourcing tasks，an improved software crowdsourcing module allocation algorithm is proposed. According to the complexity of the structure and the complexity of the technology，the complexity of the module to be developed is calculated. The cosine similarity algorithm is used to estimate the quality of the module，the importance of the module is calculated according to the critical path，and the module core degree is calculated according to the complexity，quality and importance of the module. The software crowdsourcing module allocation is realized by module core degree sorting. Experimental results show that the fitness value of the algorithm can be improved by at least 33.9，27.7 and 27.8 compared with the CFA，QFA，and KFA allocation algorithms.

【Key words】 software crowdsourcing；module allocation；cosine similarity；module core degree；fitness value

DOI： 10.19678/j.issn.1000-3428.0053768

0 概述

随着互联网的快速发展，众包服务已引起各领域的广泛关注。相对传统外包模式，众包模式[1-2]的发展为任务发布者和任务请求者提供了极大的便利。随着众包模式的发展，越来越多的用户选择采用众包模式发布软件任务，这使得对于众包软件这一课题的研究更加有意义。然而，由于众包平台的发包方和工人具有不确定性，因此如何有效准确地将多样性任务分配给能力参差不齐的工人是亟待解决的问题[3]。

在此背景下，国内外众多学者在该领域展开研究。文献[4]根据历史完成情况和当前完成情况对工人进行综合评定，提高任务分配精度。文献[5]根据工人性格对工人能力进行度量，但对于任务属性考虑不足。文献[6]从基本信息、能力、资源和描述 4 个维度建立任务模型，并基于 SIMFT 算法向工人推荐任务。该方法虽然考虑到了任务模型对于完成质量的影响，但并未深入考虑任务复杂度等影响因子。文献[7]基于多元线性回归得出影响任务质量

基金项目： 国家自然科学基金(61572371)；湖北省技术创新专项(2018ACA13)。

作者简介： 王晨旭(1997—)，男，本科生，主研方向为软件众包、服务计算；余敦辉，副教授、博士；张万山，讲师、硕士；张兴盛，本科生。

收稿日期： 2019-01-21 **修回日期：** 2019-02-22 **E-mail：** yumhy@hubu.edu.cn

的潜在因素,但该方法得出的“任务名称每增加一个英文单词,参与率降低 1.491”结论的普适性有待商榷。文献[8]得出众包任务的质量很大程度上受到任务难度影响的结论,但未考虑任务完成后可能需要维护以及用户对各任务重视程度不一致等问题,可能出现用户不满意导致返工的情况。

鉴于现有研究仍存在对模块质量、模块重要度考虑不足以及工人能力利用不充分的问题,本文提出一种基于核心度排序的软件众包模块分配算法。该算法综合考虑模块复杂度、模块质量、模块重要度,为软件模块挑选出最合适的工人。

1　问题模型

本文研究的软件众包任务均为基于竞标模式的复杂任务,相关定义及问题描述具体如下:

定义 1(模块) 模块是指已经过众包平台架构师分解的软件任务,其定义为一个四元组,$M=\{D_{\mathrm{m}},S_{\mathrm{m}},U_{\mathrm{m}},C_{\mathrm{m}}\}$,其中,$D_{\mathrm{m}}$ 表示模块的复杂度,S_{m} 表示模块的质量,U_{m} 表示模块的重要度,C_{m} 表示模块的核心度。

定义 2(软件模块网络) 软件模块网络定义为一个有向网络 $G_{\mathrm{d}}=(V_{\mathrm{d}},E_{\mathrm{d}})$,其中,结点 $v_{\mathrm{d}}(v_{\mathrm{d}}\in V_{\mathrm{d}})$ 代表软件项目中的模块,如果 2 个模块之间存在依赖关系,则构成有向边 $e_{ij}(e_{ij}\in E_{\mathrm{d}})$。

2　模块复杂度计算

模块复杂度的度量对于提升模块质量、缩短开发周期具有重要作用,本文描述的模块复杂度主要包括结构复杂度和技术复杂度[9-11]。

结构复杂度可根据软件模块网络计算得到,用来衡量软件项目中各模块之间的依赖关系,依赖关系越多,结构复杂度越高。本文主要考虑以下 4 种依赖情况[12]:

1)实现,假设 v_1 类实现了 v_2 接口,则存在有向边 $e_{12}=<v_1,v_2>$,边的权值用 w_1 表示。

2)泛化,假设 v_3 类继承了 v_4 类的功能,则存在有向边 $e_{34}=<v_3,v_4>$,边的权值用 w_2 表示。

3)依赖,假设 v_5 类的变化会引起 v_6 类的变化,则存在有向边 $e_{56}=<v_5,v_6>$,边的权值用 w_3 表示。

4)关联,假设 v_7 类和 v_8 类之间有一定的联系,该关系使得一个类知道另一个类的属性和方法,则存在有向边 $e_{78}=<v_7,v_8>$,边的权值用 w_4 表示。

考虑到上述 4 种关系在实际开发中依赖程度不同,因此本文约定 $w_1+w_2+w_3+w_4=1$。假设某个软件项目中共有 m 个模块,对于模块 i 而言其结构复杂度 f_i 为:

$$f_i=\mathrm{e}^{(-\sum w_1+\sum w_2+\sum w_3+\sum w_4)^{-1}} \tag{1}$$

考虑到实际开发中可能存在模块结构较简单,但技术实现困难的情况,为更好地度量模块的技术复杂度,本文采用专家打分的方式对模块技术复杂度进行打分(满分为 10 分),假设共有 h 名专家进行打分,对于模块 i 而言其技术复杂度为:

$$g_i=\frac{1}{10h}\sum_{j=1}^{h}score_j \tag{2}$$

其中,$score_j$ 表示第 j 个专家给出的分数。

模块复杂度 D 为:

$$D=\delta f_i+(1-\delta)g_i \tag{3}$$

其中,δ 为常数,具体数值由众包平台指定。

3　模块质量计算

考虑到模块的完成质量与工人的能力具有很强的关联性,但未进行开发之前,待开发模块的质量未知,因此需要借助已有的相似模块进行估测。本节给出基于余弦相似度算法的模块相似度度量方法、已完成模块的质量计算方法以及待开发模块的质量估测方法。

3.1　基于余弦相似度算法的模块相似度度量

余弦相似度是利用空间向量夹角的余弦来衡量对象间的差异,余弦值越接近 1,表明夹角越接近 0,2 个向量越相似[13-15]。假设历史模块集合 DB 是一个三元组,$DB=(P,A,R)$,其中,P 表示模块集,$P=\{P_1,P_2,\cdots,P_e\}$,A 表示模块的属性集合,$A=\{A_1,A_2,\cdots,A_g\}$,R 表示模块 P 的属性 A 的取值,即 $R_{ab}=\{P_a(A_b)\}$ 表示第 a 个模块的第 b 个属性的取值。

模块 p、q 的余弦相似度计算如下:

$$sim(p,q)=\frac{\sum_{i=1}^{n}(R_{pi}\times R_{qi})}{\sqrt{\sum_{i=1}^{n}(R_{pi})^2}\times\sqrt{\sum_{i=1}^{n}(R_{qi})^2}} \tag{4}$$

其中,属性 R_{pi} 取值如下:

$$R_{pi}=\begin{cases}1,\text{文本型,存在该属性}\\0,\text{文本型,不存在该属性}\\\dfrac{R_{pi}}{R_{pi}+R_{qi}},\text{数值型}\end{cases} \tag{5}$$

模块属性取值如表 1 所示。p_0、p_1 属性取值如表 2所示。p_0、p_2 属性取值如表 3 所示。

表 1　模块属性取值

模块	开发语言	软件规模	工作量
p_0	Java	180	110
p_1	Java	200	100
p_2	Python	175	90

表 2　p_0、p_1 属性取值

模块	Java	软件规模	工作量
p_0	1	0.47	0.52
p_1	1	0.53	0.48

表 3 p_0、p_2属性取值

模块	Java	Python	软件规模	工作量
p_0	1	0	0.51	0.55
p_2	0	1	0.49	0.45

p_0、p_1的余弦相似度计算如下：

$$sim(p_0,p_1)=\frac{1+0.47\times0.53+0.52\times0.48}{\sqrt{1^2+0.47^2+0.52^2}\times\sqrt{1^2+0.53^2+0.48^2}}=0.998$$

p_0、p_2的余弦相似度计算如下：

$$sim(p_0,p_2)=\frac{0+0+0.51\times0.49+0.55\times0.45}{\sqrt{1^2+0+0.51^2+0.55^2}\times\sqrt{0+1^2+0.49^2+0.45^2}}=0.332$$

经计算易知，p_0与 p_1最相似。考虑到不同属性的影响程度不同，在实际使用中，众包平台也可根据具体情况为不同属性赋予不同的权值。

3.2 已完成模块的质量计算

由于任务完成质量差导致某些模块经常需要维护且用户满意度较低，因此模块维护频率、维护时长、用户评分可以有效反映模块质量，假设 F_m 为第 k 个模块的维护频率，T_m 为第 k 个模块的维护时长，Z_k 为用户对第 k 个模块的评分（满分为 10 分），假设平台已完成 m 个模块，用 S_k 表示第 k 个模块的完成质量。

$$S_k=a\frac{\min(F_m)}{F_k}+b\frac{\min(T_m)}{T_k}+c\frac{Z_k}{\max(Z_m)} \tag{6}$$

其中，a、b、c 为权值，并且满足 $a+b+c=1,0<a,b,c<1$，该权值由众包平台根据实际情况指定。

3.3 待开发模块的质量计算

在计算待开发模块质量时，首先计算待开发模块与已开发模块的相似度，选出相似度最高的前 K 个模块，待开发模块 A 的质量计算如下：

$$S_A=e^{\left(-\sum_{i=1}^{K}s_{ki}\right)^{-1}} \tag{7}$$

4 模块重要度计算

若要缩短软件周期，则需缩短位于关键路径上的模块开发周期，因此应该尽可能为关键路径上的模块分配能力更高的工人[16]。

本文假设已经构建待开发项目的 AOE 网，假设关键路径上的结点集合 $key=\{k_1,k_2,\cdots,k_m\}$，其中每一个结点代表一个模块。根据文献[17]方法，容易计算得到关键路径，假设计算得到 n 条关键路径，获得相应的结点集合 $key_1,key_2,\cdots,key_n$，则位于关键路径上的结点集合最终为：

$$KEY=\bigcup_{i=1}^{n}key_i \tag{8}$$

对 KEY 中的结点按照开发的先后排序得到集合 KEY'，假设 KEY' 内有 V 个结点，对于第 t 个结点对应模块的重要度 U_t 计算如下：

$$U_t=L\times t+L_0 \tag{9}$$

其中，$L<0$，L_0 为初始值。

假设集合 $UKEY$ 表示非关键路径上的结点，则有：$UKEY=\overline{KEY}$，对于 $UKEY$ 中的结点 T，其对应模块的重要度 U_T 计算如下：

$$U_T=w+L_0 \tag{10}$$

其中，w 为常数，L_0 与 w 均由众包平台指定，该值可根据项目的紧急程度、发包方对平台的重要程度等因素确定。

5 基于核心度排序的软件众包任务分配

5.1 模块核心度计算

鉴于模块复杂度考虑不足时可能会将简单模块分配给高级工人导致工人能力利用不充分，模块质量考虑不足时可能将质量要求较高的模块分配给初级工人造成任务完成质量低，模块重要度考虑不足时可能会将位于关键路径上的模块分配给初级工人造成软件开发延期。为此，本文提出核心度的概念，模块核心度是模块复杂度、模块质量、模块重要度的综合体现，用以表示模块的主要特性，模块核心度 C_v 具体计算如下：

$$C_v=\alpha\times e^{\left(-\sum_{v=1}^{m}D_v\right)^{-1}}+\beta\times e^{-\sum_{v=1}^{m}S_v}+\gamma\times e^{\left(-\sum_{v=1}^{m}U_v\right)^{-1}} \tag{11}$$

其中，α、β、γ 为权值并且满足 $\alpha+\beta+\gamma=1,0<\alpha,\beta,\gamma<1$，$D_v$ 表示第 v 个模块的复杂度，S_v 表示第 v 个模块的质量，U_v 表示第 v 个模块的重要度。

5.2 软件众包任务分配算法

本节提出基于核心度排序的软件众包任务分配算法（CMA），先计算模块的核心度，将得到的模块核心度从大到小排序，再按照文献[18]方法进行工人打分并按照从高到低的顺序进行排序。鉴于本文研究重点是模块核心度，为方便描述，忽略了模块之间所需工人技能的差异。

在进行工人任务匹配时，本文采取文献[19]方法，即按照模块核心度从大到小依次分配时，每次从工人集合中选出能力最强的工人对其进行任务匹配。这样分配可以保证核心度最高的模块由技能最高的工人完成，在一定程度上可提升团队开发效率，缩短开发时间。CMA 算法具体描述如下：

输入 待开发模块集合 WM、工人集合 W、模块个数 N

输出 工人-任务匹配集合 L

1. 计算各模块的结构复杂度；
2. 计算各模块的技术复杂度；
3. 计算各模块的复杂度；
4. 找出与待开发模块相似的 k 个历史模块；
5. 计算待开发模块的质量；
6. 计算各模块的重要度；
7. 计算各模块的核心度得到核心度集合 core；
8. 将 core 内的元素从大到小排序；

9. 采用负指数模型对工人进行打分,得到工人分数集合 score;

10. 将 score 按从大到小排序;

11. for i ← |core|

12. L[i]. core = core[i];

13. L[i]. score = score[i];

14. end for

15. return L;

6　实验结果与分析

鉴于软件众包领域缺乏基准数据集,本文爬取程序员客栈和威客众包平台的 3 764 条数据,从这些数据中获取了软件的维护情况、用户对软件的评分、模块重要度,参考文献[11,18]的研究,增加了模块复杂度及工人技能评分 2 个指标,并根据 SCM 模型和工作者质量评估模型对指标值进行生成。实验在具有 2.9 GHz Inter(R)Core(TM) i7-7500U 处理器和 12 GB 内存的机器上运行,操作系统为 Windows10,编程语言为 Matlab。

6.1　算法对比

将本文提出的基于核心度排序的软件众包任务分配算法分别与以下算法进行对比:

1)复杂度优先算法(CFA)[7],即在每次分配时将复杂度最大的模块分配给能力最高的工人。

2)质量优先算法(QFA),即在每次分配时将质量最差的模块分配给能力最高的工人。

3)重要度优先算法(KFA),即在每次分配时将重要度最大的模块分配给能力最高的工人。

为满足分配需求,本文按照模块数与工人 3∶4 的比例进行分配,每次实验从数据集中随机抽取特定数量的模块进行实验,每个实验重复 10 次,实验结果取平均值,部分参数(或常数)设定如下:$\delta = 0.5$,$L_0 = 1$,$w = -0.5$,$t = 0.05$。

为比较不同算法的优劣,本文提出适配值的概念,具体如下:

$$r = \frac{1}{\sum_{i=1}^{N}((x_i - y_i) - \overline{(x_i - y_i)})} \tag{12}$$

其中,N 表示模块数,x_i 表示第 i 个工人得分,y_i 表示指标数值,该指标可以是模块复杂度、模块质量、模块重要度,$\overline{(x_i - y_i)}$ 表示一组实验中 x_i 与 y_i 差值的平均值。

由图 1 ~ 图 6 可以看出,本文算法与 CFA 算法相比质量适配值至少提升 33.9,重要度适配值至少提升 57.2;与 QFA 算法相比复杂度适配值至少提升 27.7,重要度适配值至少提升 57.9;与 KFA 算法相比复杂度适配值至少提升 29.9,质量适配值至少提升 27.8。

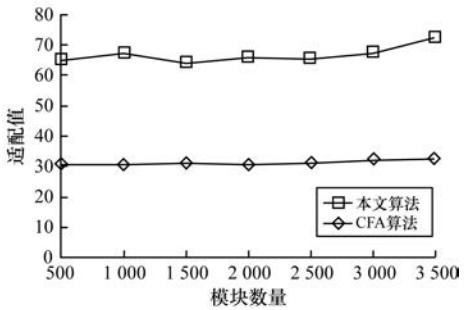

图 1　本文算法与 CFA 算法的质量适配值对比结果

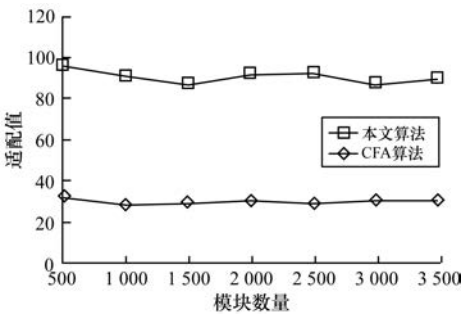

图 2　本文算法与 CFA 算法的重要度适配值对比结果

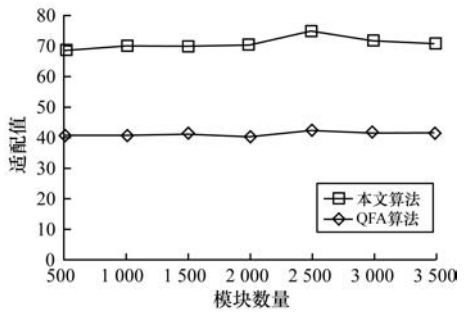

图 3　本文算法与 QFA 算法的复杂度适配值对比结果

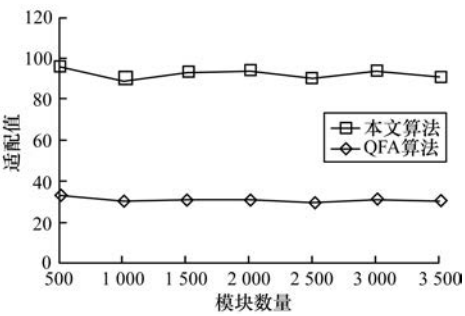

图 4　本文算法与 QFA 算法的重要度适配值对比结果

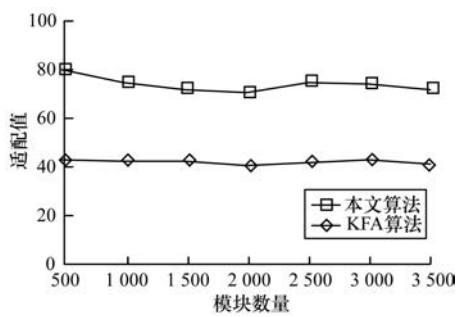

图 5　本文算法与 KFA 算法的复杂度适配值对比结果

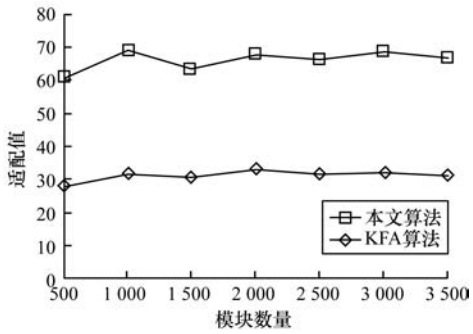

图 6 本文算法与 KFA 算法的质量适配值对比结果

6.2 参数取值讨论

对于式(6)、式(11)中的参数,综合考虑每个影响因子在实际开发中的影响效果及其约束条件,本文将 a、b、c、α、β、γ 的取值范围限定于[0.2,0.5]内,讨论参数权值组合对适配值的影响。根据图 7、图 8 并结合 $a+b+c=1$、$\alpha+\beta+\gamma=1$ 可知,当 $\alpha=0.3$、$\beta=0.5$、$\gamma=0.2$、$a=0.3$、$b=0.3$、$c=0.4$ 时适配置最大,任务分配效益最高。

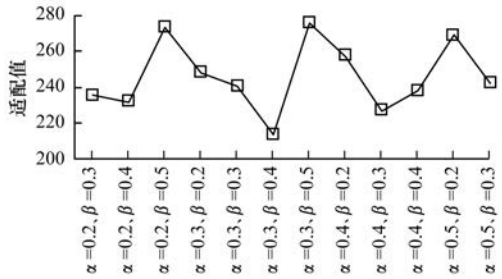

图 7 α、β 取值对适配值的影响

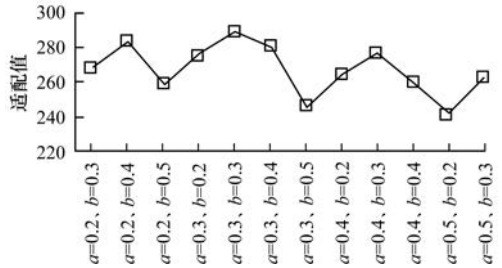

图 8 a、b 取值对适配值的影响

7 结束语

本文针对当前软件众包任务分配中模块核心度考虑不足的问题,提出基于核心度排序的众包软件模块分配算法。实验结果表明,该算法可有效提升众包工人能力利用率,众包模块开发质量,并缩短开发周期。由于本文重点研究了任务属性对任务质量的影响,对于工人和任务的匹配考虑比较单一,因此下一步可在此基础上考虑工人间的合作关系及工人对不同技能的熟练程度等,更好实现众包工人与任务的匹配。

参考文献

[1] MAO Ke,CAPRA L,HARMAN M,et al. A survey of the use of crowdsourcing in software engineering[J]. Journal of Systems and Software,2016,126:57-84.

[2] TSAI W T,WU Wenjun,HUHNS M N. Cloud-based software crowdsourcing[J]. IEEE Internet Computing,2014,18(3):78-83.

[3] 冯剑红,李国良,冯建华. 众包技术研究综述[J]. 计算机学报,2015,38(9):1713-1726.

[4] 施战,辛煜,孙玉娥,等. 基于用户可靠性的众包系统任务分配机制[J]. 计算机应用,2017,37(9):2449-2453.

[5] TUNIO M Z,LUO Haiyong,CONG Wang,et al. Task assignment model for crowdsourcing software development: TAM[J]. Journal of Information Processing Systems,2018,14(3):621-630.

[6] 赵世雄. 软件众包的开发人员和任务推荐[D]. 上海:上海交通大学,2016.

[7] 安思锦,翟健. 软件众包参与度影响因素分析及预测模型[J]. 计算机系统应用,2015,24(10):9-16.

[8] 江雨. 基于不确定任务环境的众包用户行为分析及调度策略研究[D]. 上海:华东师范大学,2018.

[9] ZHANG Haohua,FENG Wenjiang,WU Lijuan. The static structural complexity metrics for large-scale software system[J]. Applied Mechanics and Materials,2010,44-47:3548-3552.

[10] CHOWDHURY I,ZULKERNINE M. Using complexity,coupling,and cohesion metrics as early indicators of vulnerabilities[J]. Journal of Systems Architecture,2010,57(3):294-313.

[11] 付剑晶,王珂. 软件迷惑变换的鲁棒性量化评价[J]. 软件学报,2013,24(4):730-748.

[12] 何鹏,王鹏,李兵,等. 基于多粒度软件网络模型的软件系统演化分析[J]. 电子学报,2018,46(2):257-267.

[13] 许海玲,吴潇,李晓东,等. 互联网推荐系统比较研究[J]. 软件学报,2009,20(2):350-362.

[14] 李明树,何梅,杨达,等. 软件成本估算方法及应用[J]. 软件学报,2007,18(4):775-795.

[15] REN Xueli,DAI Yubiao,ZHOU Lifen. Application in effort estimation of collaborative filtering[C]//Proceedings of the 6th International Symposium on Computational Intelligence and Design. Washington D. C.,USA: IEEE Computer Society,2013:330-333.

[16] 李俊亭,王润孝,杨云涛. 关键链多项目整体进度优化[J]. 计算机集成制造系统,2011,17(8):1772-1779.

[17] 徐凤生. 一种求关键路径的新算法[J]. 计算机工程与应用,2005,41(24):82-84.

[18] DANG Depeng,LIU Ying,ZHANG Xiaoran,et al. A crowdsourcing worker quality evaluation algorithm on mapreduce for big data applications[J]. IEEE Transactions on Parallel and Distributed Systems,2015,27(7):1879-1888.

[19] PENG Xin,BABAR M,EBERT C. Collaborative software development platforms for crowdsourcing[J]. IEEE Software,2014,31(2):30-36.

编辑 陆燕菲