众包模式下基于交易成本理论的并行管理模型研究

邢 颖 郭基凤 缑西梅

(中原工学院软件学院 郑州 450007)

摘 要 众包是一种新兴的软件开发模式,具有经济学特性。针对在线交易市场中存在的问题,引入交易成本理论,设计了众包模式下的并行管理模型,并对其进行符号化定义、成本分析及均衡点设置,最后以"漏洞修复"为案例讲解了模型的应用。模型强调以市场竞争为基础的资源管理和分配,具有实际应用意义,有助于实现供需双方资源管理与优化分配,提高开发效率。

关键词 众包,交易成本,并行,模型

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.035

Research on Parallel Management Model Based on Transaction Cost Theory for Crowd-sourcing

XING Ying GUO Ji-feng GOU Xi-mei

(Software College, Zhongyuan University of Technology, Zhengzhou 450007, China)

Abstract Crowd-sourcing is a new software development model, and it possesses economic characteristics. Aiming to the problem of online trading market, this paper introduced the transaction cost theory, and designed parallel management mode for crowd-sourcing model. Then its symbol definition, cost analysis, the equilibrium point were set. Finally, for the case of "bug fixes", the application of the model was explained. Model emphasizes resource management and allocation of market-based competition, and it has practical significance, contributing to resources management and optimal allocation of both supply and demand, and improving development efficiency.

Keywords Crowd-sourcing, Transaction cost, Parallel, Model

1 引言

众包植根于一个平等主义原则:每个人都拥有对别人有价值的知识或才华,众包作为桥梁将"我"和"他人"联系起来。众包模式是指一个公司或机构把过去由员工执行的工作任务分解成小的、模块化的块,以自由自愿的形式外包给非特定的(通常是大型的)大众网络的模式[1]。众包可以完全虚拟化,企业的生产制造外包、销售外包、人力资源外包、物流外包、信息技术外包、服务外包等要求在一定的地域内实现一定的联系。众包比较适合为创新设计领域提供问题解决方案,如产品与广告设计、营销方案、技术研发、软件设计等。如今众包模式已经对一些产业产生了颠覆性的影响,如一个跨国公司耗费几十亿美元也无法解决的研发难题被几个外行人在两周的时间内圆满完成;过去要数百美元一张的专业水准图片,现在只要一美元即可买到。

众包可以抽象为在线虚拟市场。国外一些众包模式包括:TopCoder股份有限公司使用程序设计比赛来打造专业级的软件客户端外包,从个人创业到全球财富1000强企业不等。它不仅仅是一个软件众包平台,也是软件人才社区。Rent-a-Coder是连接买家和程序员的另外一个"按时间付费"的在线市场,买方是根据项目所用的时间而非最终的结果为程序员付款。iTunes App Store是为iPhone提供的基于市场方法的应用开发生态系统,内置小额支付系统[2]。应用商店表明,基于小额支付模型的软件交付是被消费者接受的,但是缺少一种让多个程序员致力于一个项目以及漏洞修复的竞标方式。综上所述,需要从根本上改变大型软件系统的方法,这些系统依赖于有机且自我调节的机制,需要探索利用市场机制的可能性来推动软件的发展,减少使用经济能够加以修复的缺陷。本文将众包任务的分解完成工作抽象为一个市场平台,通过引入交易成本理论提出新的并行管理模型,强调以市场竞争为基础的资源管理和分配,保证双方都获得最大利益。

2 交易成本理论

交易成本理论是经济学研究中较为成熟的理论,由诺贝尔经济学奖得主科斯(Coase R. H.)于1937年提出。交易成本理论的根本论点在于对企业的本质加以解释。由于经济体系中企业的专业分工与市场价格机能之间的运作,产生了专业分工的现象;但是使用市场的价格机能的成本相对偏高,而形成的企业机制是人类追求经济效率所形成的组织体。科斯与威廉姆斯的交易成本理论认为,组织经济行为的维持依赖于生产成本和交易成本之间的平衡。交易成本理论为评估不同的内部和外部组织之间的交易问题提供了一个有效的方法,为

到稿日期:2015-11-30 返修日期:2016-03-09 本文受2015年河南省科技攻关项目:众包模式下软件质量控制方法研究(152102210150)资助。

邢 颖(1985—),女,硕士,讲师,CCF会员,主要研究方向为分布式计算与云计算;郭基凤(1963—),女,硕士,教授,主要研究方向为软件工程、计算机应用;缑西梅(1970—),女,硕士,副教授,主要研究方向为软件工程、算法分析。

分析众包决策提供了一个有效的分析框架[3]，如图 1 所示。

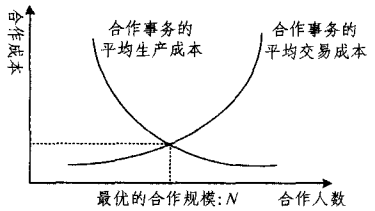

图 1 交易成本理论示意图

交易成本理论认为组织可以通过众包来降低生产成本，服务提供商通常具有规模经济带来的较低成本，当众包签约成本、管理成本、外包商供应成本之和小于自己从事的成本时，众包企业通过某种机制巧妙地将某些原本应由自己完成的工作外包给不特定的大众。人人可以表达意见与想法，企业组织集思广益，将智慧精华作为决策与运作的重要参考。企业只需付出较少的报酬即可完成任务，大大降低了成本。

3 基于交易成本理论的众包并行管理模型

3.1 设计原则

本文将众包任务的分解完成工作抽象为一个市场平台，通过引入交易成本理论提出新的管理模型。模型支持的两个基本问题是：1)所有用户关于特定工作的偏好和信息汇总；2)工作资源的优化配置。

模型的设计有以下 4 个最基本的原则。

1)自主性。其为市场工作的关键属性，所有完成工作的行为都应该由市场机制调节，这个过程绝不能被市场以外的实体控制。

2)包容性：在改进时任何提供信息和执行工作的人都应该共享奖励。

3)透明性：系统中市场资金的流动和市场工作人员执行的任务是透明的。

4)可靠性：系统应该免受操控，有强劲的抵御攻击的能力，并能防止“浅”工作。

集中控制缺乏扩展性，从而导致缺乏响应性。为了解决这个问题，设计模型时可考虑基于信誉的系统：每个贡献者都有一个等级，每个类型的任务都有一个最小等级的先决条件。等级可以由管理者进行设置，所以它可以直接给其成员权限来执行各种操作授权[4]。

3.2 模型设计

根据上文的分析，提出基于交易成本理论的并行管理模型。模型中服务被视作商品，能以“货币”的形式交易，模型由以下 3 部分组成，如图 2 所示。

1)User：消费者，即软件用户，可能是个人、公司或者其他实体。用户通过代理向市场发布自己的要求，包括所需完成任务的具体描述以及功能偏好，并报出愿意付出的价格。

2)Worker：生产者，即服务提供方。相关开发人员发布服务的价格，根据自身成本效益以及买方报价，提供解决方案，执行工作。每个任务可能都会有一系列的参与到工作流程各个部分的开发人员，例如问题的记录人员、测试的开发人员和修复的解决人员。

3)Agent：代理是模型中的核心组件，是买卖双方之间的接口，主要用于集中管理交易过程，动态管理资源配置，包含服务请求控制机制、定价机制、监控机制。

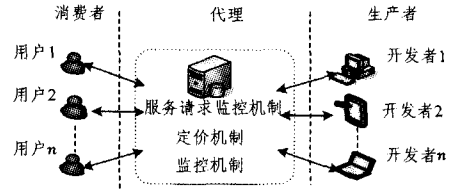

图 2 基于交易成本理论的众包并行管理模型

在市场机制中，消费者请求通常包含多种信息和偏好，例如不同种类工作的花费、时间要求、成本控制等。市场允许参与者使用合适的价格寻找供应商或消费者，供应商在市场中进行服务注册并发布资源的价格。Agent 可以接收许多用户的请求，用户有权选择提交他们的请求给不同供应商。

在多个消费者与多个供应商之间，交易资金由可信任的第三方即“代理”托管，起到交易仲裁人的作用[5]。用户购买软件时，将一部分资金放置在市场的 Agent 银行里，交易双方在分布式银行中都有账户，用户可以使用这笔资金来投标，然后开发人员(卖方)根据其成本和经济回报来进行竞标。买方从中选择他想要为此项目付款的卖方并且把资金交给 Agent 保管，市场达到均衡点，达成协议。当工作完成后，买方放在代理的资金付给卖方，谈判成功。竞标在 Agent 中有一个截止日期，如果竞标到期，那么这些“钱”会返回给投标人。

模型强调以市场竞争为基础的并行资源的管理和分配，利用供求关系对资源所有者和使用者进行动态配置，试图得到一个所有资源的均衡点，保证双方都获得最大利益。

3.3 符号化定义

本文采用基于市场的机制来推动软件的发展，可集中解决程序资源所存在的问题，这是一个动态配置的过程。根据上文分析，将模型参与的实体进行符号化定义，主要包括以下基本实体。

U：消费者，即软件用户，可能是个人、公司或者其他实体。

J：任务，即用户想要通过众包平台实现软件中的相关工作。

W：生产者，即服务提供方。相关开发人员提供解决方案，执行工作。

K：类别，用于对工作进行分类，如正确性、功能、安全性、测试等。

L_{kj}：如果任务 j 里面有种类 k，那么其值是 1，否则是 0。

R^t_{uj}：报价，在时间 t 中用户 u 愿意为解决工作 j 提供的报酬。

C^t_{wj}：成本，在时间 t 中开人员 w 执行工作 j 的实际花费。如果工作人员 w 在时间 t 中没有能力或者对执行的工作 j 不感兴趣，那么认为 $C^t_{uj}=\infty$。

货币可以是真实的，也可以是虚拟的。标签用于分类，并且允许计算关于不同类别的工作项目的汇总统计。市场对这些类别是不可知的，用户可通过对特定种类的工作提供回报来表达偏好。

3.4 成本分析与托管

市场提供资金的方式依赖于软件正在使用的商业模式，

不管系统是开源的还是闭源的，市场必须设计成不存在“下游依赖性”[6]。用户购买服务时，将一部分资金放置在市场的 Agent 里，然后用户可以使用这笔资金来投标，如果用户在一段时间内不投标，那么这些钱会返回给卖方。

根据分析，在时间 t 内工作 j 的总报价：

$$R_j^t=\sum_{u\in U}R_{uj}^t \tag{1}$$

因此可定义“用户总需求”为软件每一组成部分可提供报酬的总和：

$$R^t=\sum_{j\in J}R_j^t \tag{2}$$

当式(2)中的 $R=0$ 时，说明对于软件没有任何更改或者添加的需求。

本文表示在时间 t 中工作的特定种类 k 的需求是：

$$R^t[k]=\sum_{j\in J}R_j^t\cdot L_{kj} \tag{3}$$

市场中在时间 t 内把正确性需求描述为 $R^t[correctness]$。如果正确性需求是 0，意味着软件中没有缺陷，类似地可以量化需求，例如安全性、新的功能和支持特定平台等。

3.5 均衡点

每个任务中都可能有参与到工作流程各个部分的工作人员，例如，问题的记录人员、测试的开发人员和修复的开发人员。一般来说，工作完成后，所有的贡献者可以共享工作所得到的报酬；付款也可以在一段时间后进行，以确保贡献的公平性。由于报酬在开发者之间可以进行动态分配，一般来说，服务提供方选择利益大于零的任务进行开发，即报酬高于本身付出的成本。

更具体地，可以定义在时间 t 内对于工作 j 的价值为：

$$P_j^t=\max_{w\in W}(R_j^t-C_{wj}^t,0) \tag{4}$$

即工作的价值为以最低的成本执行它所获得的报酬的净成本。

本文定义整个工作市场的潜在价值为：

$$P^t=\sum_{j\in J}p_j^t \tag{5}$$

在时间 t 内，一个系统的正确性需求潜力为 $P^t[correctness]$，$P^t[correctness]\leqslant R^t[correctness]$。

用户需求 R_j^t 被认为是在工作 j 中按照用户需求工作的价格(或者买入价)，C_{wj}^t 代表工作人员 w 愿意为工作 j 提供工作的价格(或者卖出价)。理想情况下，每当买入价高于或等于卖出价时，市场就促使工作完成。所以，当 $P^t[correctness]=0$ 时定义一个系统是正确性平衡。如图 3 所示，E 点即是均衡点。

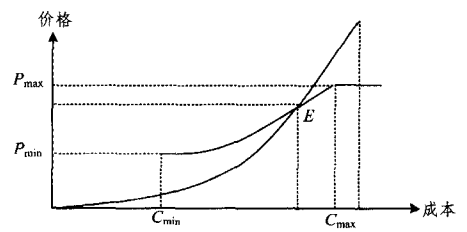

图 3　交易均衡点示意图

实际的工作环境是动态的，工作人员对每一项工作都可以在解决方案质量和执行工作的成本之间达到最佳平衡。例如对于正确性 $P^t[correctness]=0$；系统会不断地向平衡移动，通过执行来自市场竞争的潜在价值的工作来向前推动[7]。尽管一个绝对的平衡在现实中可能永远不会实现，但是均衡点的定义在系统行为框架方面仍然是十分有用的。

4 案例分析

4.1 工作流程

本文以某个用户希望通过众包平台进行软件“漏洞修复”为案例来介绍模型的实际应用：开发人员修复漏洞，然后测试人员对修复程序进行测试。

某软件“漏洞修复”的工作流程如下：

步骤 1　报告。用户提交错误报告，可以是当应用程序崩溃时简单地点击弹出框中的按钮(“发送报告给提供商”)或者提交漏洞数据库中明确存在的问题。

步骤 2　竞标。对于修复该漏洞，通过代理以报酬 R_j 进行竞标。

步骤 3　组织整理。漏洞被确定为工作流程里的一个独立部分，确定等级。

步骤 4　重现。创建和提交重现错误的方式，从而让修复者可以开始工作。

步骤 5　修复。制定漏洞修复，普通等级的漏洞可以修复，高等级的漏洞需要往上层提交。

步骤 6　测试。测试修复，要么通过提供可执行的测试用例，要么通过运行系统。后者因为需要较大程度的信任，所以通常需要很高的等级。通过修复，测试获得更高的奖励。

步骤 7　提交。提交修复到程序规范代码库，通常需要很高的等级。

步骤 8　发布补丁。漏洞修复完成后发布新版本，投标者可以指定支付当前补丁或等待新版本。

4.2 模拟分析

本节对交易过程进行分析，假设漏洞发现者为 W_1，漏洞修复者为 W_2，漏洞测试者为 W_3。

在时间 T_0 用户 U 报告了一个漏洞 B；开发者(公司、组织或者开源社区)在 Agent 中发布关于 B 的相关信息；用户开始对 B 进行付费以反映修复的价值，在一段时间 T_1 内，奖励随着这个漏洞上升为 $R_B^{T_1}$；开发人员认为充足的钱可以促成修复 B 盈利，就会进入并且开始对 B 工作；在时间 T_2 后，报酬为 $R_B^{T_2}$，开发人员完成了工作，其他用户或者测试者通过测试软件来检查修复是否健壮；如果在时间 T_3 内漏洞在开发人员提交的解决方案中被发现，那么负责漏洞的检测人员分享奖励资金 $R_B^{T_3}$。

根据本文提出的模型，并行交易处理方式如图 4 所示。

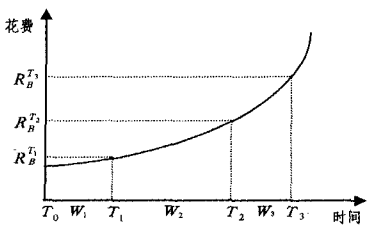

图 4　软件漏洞修复案例交易图

可以认为支付方案在漏洞发现者 W_1、漏洞修复者 W_2 和漏洞测试者 W_3 三者之间以不同的比例分配奖励资金 R_B，以反映他们解决相应漏洞的相对贡献力度。例如，漏洞的发现

者被确定为第一个发现特定漏洞的人，该漏洞需要被证实为真实的。确定该漏洞被初始发现时的重要性比例通过$(T_1-T_0)/(T_3-T_0)$进行计算，若该比例较高，则说明此发现十分有用，因为在很短的时间内就能够累积一大笔资金。很明显，部分用户群将会从修复漏洞中获得重要的实用程序。

同样，如果花费的时间段T_2-T_1较长，则往往表明修复意义重大，并且漏洞修复者花费了相当大的精力。同样，时间段T_3-T_2越长，修复就越健壮。因此，测试者越早发现其中的缺陷，越能从中获得更大的份额。

结束语 本文将众包任务的分解完成工作抽象为一个市场平台，引入交易成本理论，设计了众包模式下的新的并行管理模型，然后分析了设计原则，并用数学方法进行符号化定义、成本分析与托管以及买卖双方均衡点的设置，最后以“漏洞修复”为案例讲解了模型的应用。模型具有一定的创新性，有助于实现供需双方资源管理与优化分配，提高开发效率。“代理”监管的有效性以及模型中的效率量化是下一步要研究的内容。

参 考 文 献

[1] ZHANG Z Q, et al. Research on quality control strategy and evaluation algorithm of [J]. Chinese Journal of Computers, 2013, 36(8): 1636-1649. (in Chinese)
张志强，等. 众包质量控制策略及评估算法研究[J]. 计算机学报，2013，36(8)：1636-1649.

[2] ITWORLD B A. Finding freelance jobs: 6 sites for talented techies [OL]. http://www.itworld.com/article/2826288/careers/finding-freelance-jobs-6-sites-for-talented-techies.html.

[3] http://baike.so.com/doc/6347317-6560942.html

[4] BUYYA R, ABRAMSON D, GIDDY J, et al. Economic models for resource management and scheduling in grid computing [J]. Concurrency & Computation, 2002, 14(13-15): 1507-1542.

[5] CHE Z B, XING Y. Research on Composable Cost Model of Cloud Resource Economics Management Structure [J]. Computer engineering and Science, 2011, 33(8): 185-189. (in Chinese)
车战斌，邢颖. 云资源经济学管理架构中组合成本模型研究[J]. 计算机工程与科学，2011，33(8)：185-189.

[6] TRUONG H L, BRUNNER P, NAE V, et al. Dipas: A distributed performance analysis service for grid service-based workflow [J]. Future Generation Comp. Syst., 2009, 25(4): 385-398.

[7] GEIMER M, SHENDE S, MALONY A D, et al. A generic and Configurable Source-code Instrumentation Component[M]// Allen G, Nabrzyski J, Seidel E, et al., eds. ICCS (2), Vol. 5545 of Lecture Notes in Computer Science. Springer, 2009: 696-705.

(上接第 143 页)

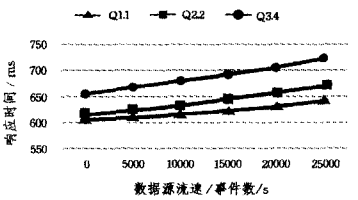

图 4 各案例的响应时间对比

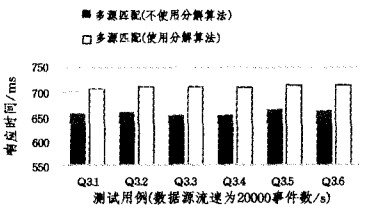

图 5 多源匹配使用分解算法前后的响应时间对比

结束语 本文提出了一种新式的语言 CEStream 和与之相对应的查询引擎，并针对分布式环境进行了优化，有效地控制了系统各个节点之间的通信量，充分利用集群的计算能力，提高了整体性能。CEStream 语言的设计与实现仍需要继续研究与发展。在语言设计中可以考虑引入更多实用的语言功能。

参 考 文 献

[1] ZHANG Q, CHENG L, BOUTABA R. Cloud computing: state-of-the-art and research challenges [J]. Journal of Internet Services and Applications, 2010, 1(11): 7-18.

[2] ZENG K, YANG M, MOZAFARI B, et al. Complex pattern matching in complex structures: The XSeq approach [C]// 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013: 1328-1331.

[3] BUGHIN J, CHUI M, MANYIKA J. Clouds, big data, and smart assets: Ten tech-enabled business trends to watch [J]. McKinsey Quarterly, 2010, 56: 75-86.

[4] BAI Y, THAKKAR H, WANG H, et al. A data stream language and system designed for power and extensibility [C]// CIKM. ACM, 2006.

[5] CUGOLA G, MARGARA A. Processing flows of information: From data stream to complex event processing [J]. ACM Computing Surveys (CSUR), 2012, 44(3): 1-62.

[6] CUGOLA G, MARGARA A. RACED: an adaptive middleware for complex event detection [C]// Proceedings of the 8th International Workshop on Adaptive and Reflective Middleware. ACM, 2009: 5.

[7] Peng F, Chawathe S S. Xpath queries on streaming data [C]// SIGMOD. 2003.

[8] DIAO Y, IMMERMAN N, GYLLSTROM D. Sase+: An agile language for kleene closure over event streams [OL]. [2012-12-23]. http://archive.systems.ethz.ch/www/dbis.ethz.ch/education/ws0708/adv_top_infsyst/papers/sase_tr07.pdf.

[9] ABADI D, CARNEY D, CETINTEMEL U, et al. Aurora: a data stream management system [C]// SIGMOD. ACM, 2003.

[10] RASANEN O, KAKOUROS S. Modeling Dependencies in Multiple Parallel Data Streams with Hyper dimensional Computing [J]. Gnal Rong Lr, 2014, 21: 899-903.

[11] AGUADO J, MENDLER M. Towards Strategies for Data Flow Programming [C]// Preproceedings of the 22nd Symposium on Implementation and Application of Functional Languages (IFL 2010). 2010: 372-473.

[12] KIEBURTZ R B. Codata and comonads in Haskell [J/OL]. http://core.ac.uk/display/24270653.