中国科学 信息科学
SCIENTIA SINICA Informationis

Crowd-intelligence-based software development method and practices

王涛, 尹刚, 余跃, 张洋 and 王怀民

Citation: 中国科学: 信息科学 **50**, 318 (2020); doi: 10.1360/SSI-2019-0187

View online: http://engine.scichina.com/doi/10.1360/SSI-2019-0187

View Table of Contents: http://engine.scichina.com/publisher/scp/journal/SSI/50/3

Published by the 《中国科学》杂志社

中国科学: 信息科学 2020 年 第 50 卷 第 3 期: 318–334
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
群体智能专题 · 论文

基于群智的软件开发群体化方法与实践

王涛[1,2*], 尹刚[2], 余跃[1,2], 张洋[1,2], 王怀民[1,2*]

1. 国防科技大学计算机学院并行与分布处理重点实验室, 长沙 410073
2. 复杂系统软件工程湖南省重点实验室, 长沙 410073
* 通信作者. E-mail: taowang2005@nudt.edu.cn, whm_w@163.com

收稿日期: 2019–08–29; 修回日期: 2019–11–12; 接受日期: 2019–12–23; 网络出版日期: 2020–03–05

国家重点研发计划 (批准号: 2018AAA0102304) 和国家自然科学基金 (批准号: 61702532) 资助项目

摘要 互联网技术的发展对软件开发技术、运行形态和服务模式都产生了前所未有的影响, 以开源和众包为代表的大规模群体协作实践所蕴含的群体智能机理为网络时代的软件开发带来重大启示. 本文以开源和众包大规模实践为案例, 深入分析了开源创新和众包生产模式, 凝练提出了以大众化协同、开放式共享和持续性评估为核心的群智软件开发机理, 从群体协作基础环境、群体协作机制模型和群体协作支撑技术 3 个方面, 深入讨论了基于群智的群体化软件开发服务环境涉及的关键要素以及我们的开源实践, 并提出了群智软件开发未来面临的重大挑战, 希望能为网络环境下基于群智的群体化软件开发提供有益的视角和借鉴.

关键词 开源, 众包, 群智, 群体化开发, 大众化协同, 开放式共享, 持续性评估

1 引言

互联网的发展对软件开发技术、运行形态和服务模式都产生了前所未有的影响 [1], 软件逐渐从面向传统领域的、以计算机为中心的信息化载体, 发展成为以网络为中心、与经济社会文化创新性融合的国家性甚至全球性基础设施, 软件系统的复杂度持续增长、需求持续变化、系统持续演化, 软件开发的关注点从过去研发相对独立的软件产品向构建多种元素相互依赖持续演化的软件生态转变, 经典的自动化和工程化软件开发方法局限性日益凸显 [2,3].

兴起于 20 世纪 90 年代的开源软件, 通过互联网将分布于世界各个角落的软件开发者和用户等联接起来, 在少数核心人员的协调下, 利用业余时间共同协作打磨出诸如 Linux, Hadoop 等众多商业级软件产品, 获得了巨大的成功 [4]. 2006 年, Howe [5] 在 *Wired Magazine* 上首次提出了众包的概念, 其核心思想是通过互联网将过去由公司或者机构中特定雇员执行的工作任务以自由自愿的方式外包给

引用格式: 王涛, 尹刚, 余跃, 等. 基于群智的软件开发群体化方法与实践. 中国科学: 信息科学, 2020, 50: 318–334, doi: 10.1360/SSI-2019-0187
Wang T, Yin G, Yu Y, et al. Crowd-intelligence-based software development method and practices (in Chinese). Sci Sin Inform, 2020, 50: 318–334, doi: 10.1360/SSI-2019-0187

大范围的开放大众去完成, 并在 Amazon 的 Mechanical Turk 平台、海量图像识别等领域得到大规模应用和实践.

开源与众包模式在不同领域取得了巨大的成功, 两者在本质上都是对大规模群体协作中开放共享与群体协同核心理念的具体实践, 对互联网环境下的软件开发产生了深刻的影响: 从生产者的角度, 大规模的软件开发使业余爱好者由被动的用户变为主动的参与者; 从生产工具的角度, 软件开发工具由过去的昂贵且不智能变得更加平民化和智能化; 从生产资料的角度, 由过去组织内部封闭独占的软件资产扩展到全互联网开源共享的智力作品; 从生产组织的角度, 由封闭大企业的强组织生产管理模式下的软件生产向开放大社区的弱组织创新引导模式下的协同创作转变. 这种变化的本质是由过去的专业人员完成工业化软件生产向大众群体共同参与、软件创作与生产相融合的转变.

分析和研究开源与众包实践背后所蕴含的基于互联网的大规模群体协作核心机理, 将其中涌现的自主的群体协作行为转化为自觉的群体化协作模型, 形成可预期的群智涌现以及基于群智的软件开发生态, 将为解决经典软件开发方法所面临的难题提供新的途径和机遇 [6].

本文以开源和众包实践为研究对象, 深入分析基于群智的大规模群体协作模型和关键机制, 探讨基于群智的群体化开发环境, 以期为互联网环境下的软件开发提供有益的理论指导和技术支持. 本文内容组织如下: 第 2 节分析介绍了开源与众包; 第 3 节深入阐释了群体化开发方法的大众化协同、开放式共享和持续性评估三要素; 第 4 节从群体协作基础环境、机制模型以及支撑技术 3 个层面详细阐述了基于群智的群体化软件开发环境; 第 5 节介绍了开源软件开发中的群体化实践; 第 6 节总结全文并展望未来工作.

2 开源与众包

开源软件是指一种源代码可以自由获取、修改和传播的计算机软件, 其发展是一个群体参与不断扩大、组织协同不断演变的过程. 早期的开源主要是黑客团体之间朴素的、下意识的行为, 参与者之间通过邮件等工具进行协同. 随着个人计算机的快速发展, 以 Richard Stallman 为代表的开源倡导者则发起了自由软件运动, 发明了 GPL 等自由软件许可证, 试图保护开发者创新自由. 这一阶段的开源开始吸引大量专业开发者的参与, 并出现了 Apache, Linux 等开源社区. 为了更好地平衡开放理念和商业利益, 业界开始探索开源新模式, 并在 1998 年的开源峰会上提出了 “开源软件” 的概念和定义. 开源软件奉行更为宽容和友好的许可证制度, 因此得到更多的支持, 吸引了大量专业开发者、业余爱好者以及商业公司的加入, 开源运动进入迅猛发展阶段. 随着开源软件的不断发展, 越来越多的商业软件公司开始认识到开源模式在提高软件创新能力和降低软件成本等方面的巨大潜力, 通过建立开源社区、参与开源开发甚至主动贡献开源项目等方式, 更为积极主动地参与开源, 引导开源社区, 探索平衡开放理念和商业利益的新模式以及群体协作新机制 [7], 开源软件与商业软件进入融合发展的新阶段. 开源软件在发展过程中逐渐形成了一种利用大众群体智慧进行软件开发的开放协作开源模式, 实现了高效率高质量的创新软件开发 [8].

众包是一种分布式的问题解决模式, 即公司或者机构通过互联网将任务公开发布出去, 利用大范围志愿者的创意和技能去解决问题 [9]. 众包模式的基本流程一般由管理者 (任务请求者) 首先将问题分解为细粒度的任务, 然后通过互联网对外公开发布, 在一定激励机制的驱动下, 大量具有相应技能的志愿者群体 (任务完成者) 申领任务、协同或者独立完成并提交给任务请求者, 任务请求者汇聚和评估任务完成者的结果, 实现问题的解决. 众包根据大众参与的形式可以分为协作式众包和竞赛式众包. 协作式众包所需要解决的任务通常是需要参与者共同协作才能完成, 且完成任务的大众通常没有物质

的奖励回报, 竞赛式众包所需解决的任务通常是由个人或者小团体独立完成, 然后通过竞争方式评估任务最佳解决方案且完成任务最佳的个人或者团体会得到相应的奖励. Amazon 的 Mechanical Turk 平台以及 InnoCentive 平台都是典型的通用性任务众包平台 [10,11]. 在软件开发领域, 众包模式也得到大规模应用. 其中, 开源开发本身就是一种协作式众包形式. 此外, 像 TopCoder 等则是软件开发领域的典型竞争式众包平台, 将软件开发任务以众包形式发布并提供一定的物质奖励, 众多参与者提交代码后由平台进行评分, 得分排名靠前的参与者将获得相应的物质奖励, 通过这种方式实现全球优秀开发群体的汇聚以及软件开发任务的高质量解决 [12].

开源与众包的核心思想是一致的, 即通过互联网汇聚和联接大规模的志愿参与者, 借助群体智慧的力量, 通过协作完成单独依靠个人或者计算机难以完成的任务 [13]. 但是, 两者之间也存在较大的差异性和互补性.

- 在模型出发点方面, 开源源于参与者对知识共享和技术创新的追求以及个人能力或者声誉的提升等, 而非直接的物质收益, 更多的是一个协同创作模型; 众包则源于企业降低问题解决成本的实际需求, 参与者则主要以物质收益为目标, 更多的是一个商业生产模型.
- 在大众参与机制方面, 开源最初主要依靠参与者的兴趣爱好和个体需求来吸引大众自愿参与, 近年来越来越多的商业公司安排全职员工参与开源 [14]. 但是开源模式目前对自由参与者尚缺乏足够的激励机制, 对参与者创新利益的保护不足; 众包则通过采用一系列激励机制, 吸引开放大众的主动参与, 并通过奖励回报实现对参与者利益的保护.
- 在任务特点方面, 开源需要解决的通常是问题边界不清楚、难以有效细分的复杂创新型任务, 这一类任务通常具有动态变化、多因素交织的特点; 众包擅长解决的则是规模较大但是边界清楚, 可以分解为大量独立子问题的生产性任务, 这一类任务通常具有重复性或者机械性的特点.
- 在任务解决方面, 开源软件开发不是一次性, 而是一个持续迭代演化的过程, 除非软件停止开发和运维, 否则任务就不会停止, 参与大众之间以协作为主, 竞争为辅, 在相应机制的支持下共同协作完成; 众包任务通常是一次性的, 有明确的截止期限, 参与大众之间以竞争为主, 协作为辅, 完成任务后通过竞争方式产生最优解决方案并获得回报 [15].

开源与众包是利用群智解决大规模复杂问题的典型实践. 软件开发是一项创造性和生产性相融合的复杂问题, 如何有效借助群智来解决, 如何在软件开发过程中实现持续稳定、可预期的群智涌现则需要相应的机制和技术的支撑.

3 基于群智的群体化开发方法

开源和众包的成功实践表明, 通过互联网联接起来的大规模群体, 在特定环境支持下能够爆发远超个体能力的群体智能 [16], 给网络时代的软件开发带来了重大启示. 基于群智的群体化开发方法将开源与众包的群体协作思想与机制和传统软件开发的工程化生产思想与机制结合起来, 包括 3 项核心要素: 大众化协同、开放式共享和持续性评估.

- 大众化协同. 自由松散的开源和众包活动释放出了惊人的生产力, 大规模外围群体的参与是重要基础. 互联网环境下参与边界的开放性以及协作任务的分解极大地降低了参与门槛, 使得参与群体高度异质, 真正成为大众化参与的群体协同. 将传统面向 “专业程序员” 扩展到面向 “多样化群体”, 实现外围群体与核心团队之间的大众化协同是群体化开发方法的核心要素之一.
- 开放式共享. 资源共享与复用是提高软件开发效率和质量的重要途径. 互联网环境下可共享资源不再处于封闭、静止状态, 而是存在结构异质、来源多样、高度分散且快速增长的特点. 将互联网

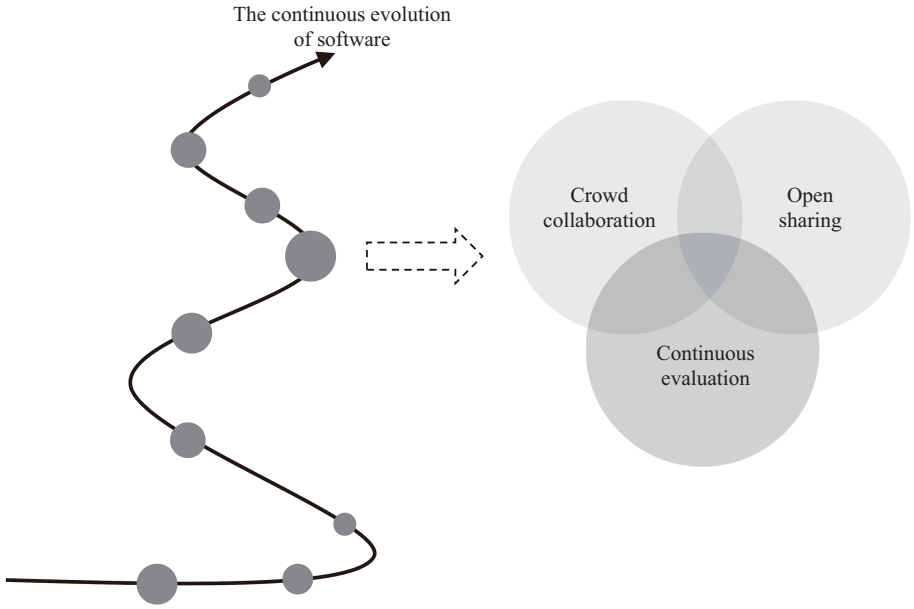


图 1 软件迭代演化中的群体化方法

Figure 1 The crowd methods in continuous evolution of software

上 “碎片化、无序” 的原生资源转变为 “聚合、有序” 的共享资源, 实现大范围的开放式资源共享, 是群体化开发方法的核心要素之二.

- 持续性评估. 群体化软件开发是一个不断迭代演化的过程, 对开发过程和参与实体的持续准确评估是确保软件向预期方向演化的重要保证. 互联网环境下的参与个体、共享资源以及协作任务都处于动态演化中. 将传统的单一维度的静态分析转变为多个维度的动态量化度量, 实现对软件开发参与实体的持续性评估是群体化开发方法的核心要素之三.

在群体化开发过程的不同阶段都需要高效的群体协同、资源共享与目标评估. 群体化开发过程包括参与个体、协作任务和协作资源 3 类参与实体, 大众化协同、开发式共享和持续性评估围绕这 3 类参与实体相互交织并随软件的迭代演化不断发展, 如图 1 所示.

3.1 面向协作任务的大众化协同

软件开发是一种智力密集型的群体协同活动 [17], 群体协同机制对软件开发效率和质量有重要影响. 群体协同既存在于核心参与者之间, 也发生在核心参与者和大众参与者之间. 参与协作的群体围绕任务建立直接或间接关联, 形成以任务为中心的大众化协同, 如图 2 所示.

群体化软件开发需要实现软件创作与软件生产的有机结合. 其中, 软件创作通常是由创作群体的灵感和创意驱动, 而软件生产则是由预先定义明确的用户需求驱动. 类似其他群智协作领域, 根据问题和求解方法的不同, 群体化软件开发的创作与生产过程通常包括 3 类典型任务:

第 1 类是问题表述完全清楚, 即不仅能够给出求解结果的逻辑判断, 而且还可以给出求解问题的分拆方法或分拆结果, 群体协同需要解决的是分拆后的子问题, 软件回归测试是典型的此类任务;

第 2 类是问题表述清楚, 即能够给出求解结果的逻辑判断, 但求解问题的 (分拆) 方法或 (分拆) 结果还不清楚, 群体协同需要解决的是问题求解方法或者分拆问题的方法, 软件缺陷修复是典型的此类任务;

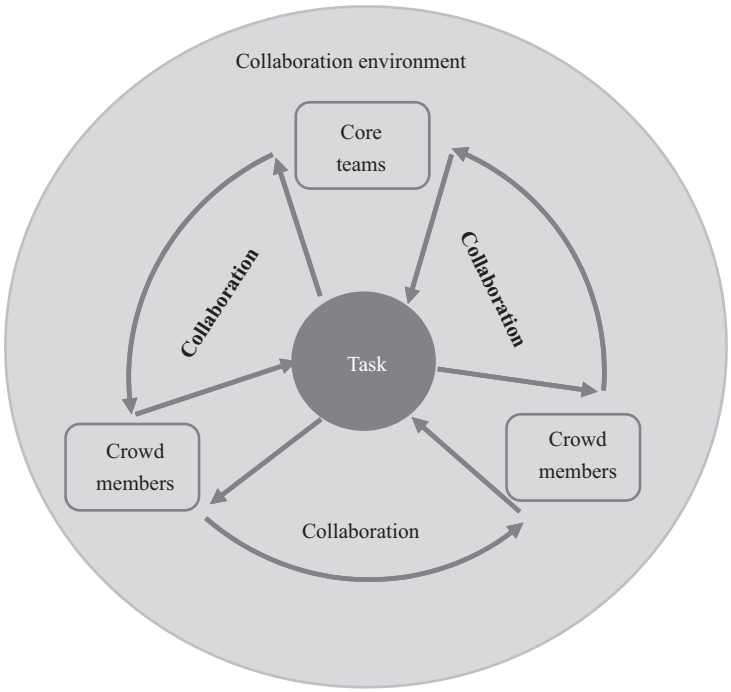


图 2　面向协作任务的大众化协同

Figure 2　Collaborative task oriented crowd collabotation

第 3 类是问题表述并不清楚, 即不能给出求解结果的逻辑判断, 不同的人对问题及其结果可能产生不同的理解, 但人们对问题的理解有共同基础, 对不同理解的部分能够设法交流并相互影响, 群体协同需要完成的任务是对问题的共同理解或清晰表述, 软件创意需求的获取是典型的此类任务.

针对不同的任务特点, 需要相应的群体协同模型. 2016 年 Michelucci 等 [18] 科学家在 *Science* 上发表文章对群体智能进行了深入分析, 提出了任务分解模型、工作流模型以及问题求解生态系统模型 3 类群体协同模型, 对群体化软件开发中的大众化群体协同具有重要参考意义. 其中, 任务分解协作模型主要针对具有大量需要重复性处理且处理规则和处理过程相对简单的子任务的一类问题, 参与群体对等, 个体之间的协作过程简单; 工作流协作模型主要针对定义清楚但是难以拆分, 且子任务间存在关联依赖的一类问题, 参与群体具有不同角色, 个体之间的协作过程较为复杂; 问题求解生态系统模型主要针对动态变化、多因素交织且需要参与者高度协调才能完成的问题, 参与群体角色多样化, 个体之间围绕任务经历多次迭代演化, 协作过程非常复杂.

在群体化软件开发过程中, 三类问题广泛存在且相互交织, 某类问题拆分后的子问题可能变成了另外一类典型问题, Michelucci 等总结的协同模型给出了重要参考, 针对群体化开发过程中的不同问题可以借鉴单个或者多个模型来协同解决.

3.2　面向协作资源的开放式共享

软件开发是一个多样化群体协作的过程, 涉及到软件代码、知识信息等大量不同资源的分享 [19]. “敝帚自珍” 式的资源封闭独占会导致协作群体内大量的重复劳动, 极大地降低群体协作效率. 核心参与者与大众参与者通过互联网进行资源的主动发布和便捷获取, 实现协作群体内资源的透明化和开放式共享, 将有效地降低群体协作开销, 提高群体协作效率, 如图 3 所示.

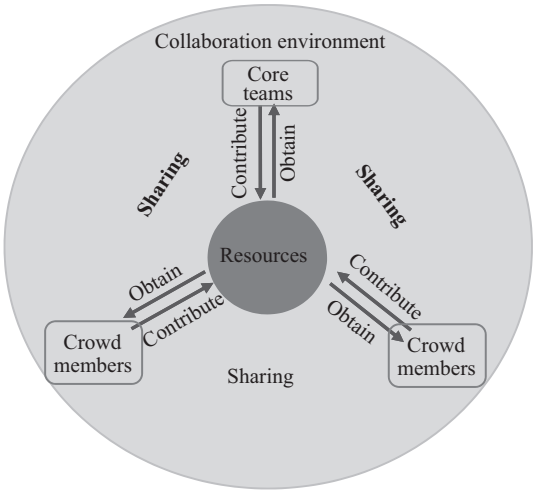


图 3 面向协作资源的开放式共享

Figure 3 Collaborative resource-oriented open sharing

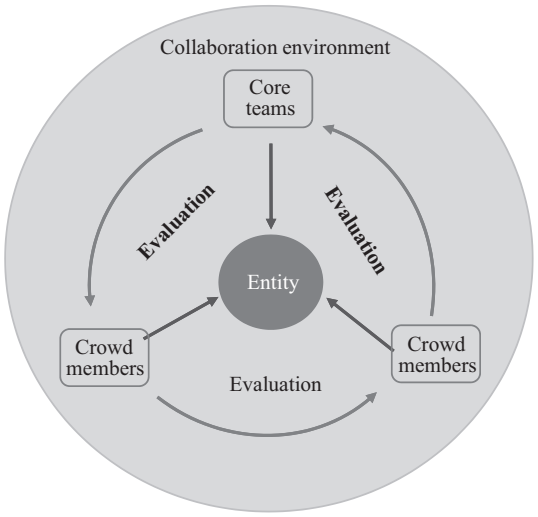


图 4 面向协作实体的持续性评估

Figure 4 Collaborative entity-oriented continuous evaluation

互联网技术的发展为资源的开放式共享提供了基础条件. 协作开发过程中产生的软件源代码、缺陷报告等过程数据, 软件运行日志、软件维护记录等运维数据, 软件技术报告、软件使用评论等反馈数据等都以前所未有的透明方式通过互联网发布出来, 任何参与者都可以通过互联网免费获取协作开发所需要的共享资源.

但与此同时, 这些共享资源也呈现出高度分散、类型多样、高速增长、不断演化的特征, 资源的高效分享需要解决资源的发布、管理与获取等问题. 群体化开发中的资源共享需要认知到 “互联网即资源库” 的理念, 将传统软件构件库的 “封闭、静态” 个体管理模式转变为网络条件下的 “开放、动态” 的群体汇聚模式, 为互联网上 “无序、分散” 的原生态软件资源提供 “有序、聚合” 的资源组织模式, 实现以资源为中心的开放式共享.

3.3 面向协作实体的持续性评估

软件开发是一个持续迭代演化的过程 [20], 这一过程中的协作实体, 主要包括参与个体、协作任务以及协作资源, 同样都处于动态变化过程中. 脱离演化过程的静态和孤立的分析将导致片面、不准确的判断, 需要从联系和发展的视角对参与实体进行持续性评估, 确保群体化开发向预期目标演化, 如图 4 所示.

- 面向协作参与者的持续性评估. 群体协作的参与者个体之间在文化背景、专业技能等方面都存在巨大的差异性, 这种差异性会对群体协作产生很大的影响 [21]. 同时, 这些参与者并非一成不变, 而是随着参与时间、参与环境等不断变化的. 通过对参与者的协作历史、跨社区的协作行为以及不同的行为模式等进行持续的关联分析, 将传统仅考虑技术能力的单一维度扩展为涵盖技术能力、个人意愿、管理能力等的多维度的参与者评估模型, 从而对参与者进行准确的角色定位和任务分派.
- 面向协作任务的持续性评估. 软件开发过程中的协作任务通常是不断迭代的, 且不同协作任务之间紧密联系相互影响, 对协作任务的持续性评估是确保项目开发顺利推进的重要保证 [22]. 开发者和用户等对协作任务的反馈是进行有效评估的重要数据来源, 通过大众反馈结合专家知识可实现对任务完

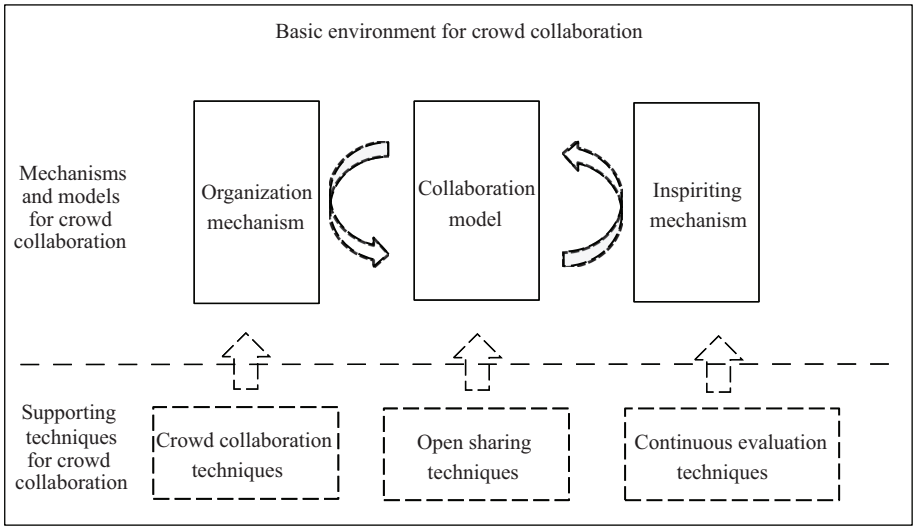


图 5　基于群智的群体化开发环境

Figure 5　Crowd-intelligence-based development environment

成情况、任务优先级以及后续任务规划等进行持续有效的评估.

• 面向协作资源的持续性评估. 共享的协作资源具有来源广泛以及高度碎片化等特征, 其质量和可信度对开发质量和效率具有很大的影响, 需要进行持续有效的评估 [23]. 一方面, 这些资源在发布共享的过程中已经积累了大量的使用和评价信息, 基于这些数据构建基于大众反馈的评估模型; 另一方面, 相同的协作资源可能存在不同的来源和表述, 通过资源溯源以及多源对比等方法实现对协作资源的持续性评估. 例如, OSSEAN 平台通过持续和实时汇聚全球开源社区关于开源项目的数据, 改变传统从软件质量这一单一维度的评估方法, 通过关联分析实现对开源资源包括社区流行度、开发活跃度、项目健康度和团队健康度等的多维度评估, 进而提供不同类别开源资源的实时排行 [24].

4　基于群智的群体化开发环境

群体化开发需要将自由动态、高度分散的大规模群体有效组织起来, 实现高效协作, 离不开相应环境的支撑. 本文从支撑群体化软件开发的视角, 围绕基础环境、机制模型和支撑技术 3 个层面对当前相关研究和实践进行梳理和总结, 本文将其概括为基于群智的群体化开发环境, 如图 5 所示.

基于群智的群体化开发环境主要包括群体协作基础环境、机制模型以及支撑技术 3 部分. 其中, 协作基础环境是面向所有项目, 协作机制模型则是面向具体项目或项目集, 协作支撑技术提供基础技术支撑, 三者共同形成了软件的群体化开发环境, 实现对大规模群体协同创作与生产的支持.

• 群体协作基础环境. 在大规模群体协作中, 每一个参与者都处于一个开放互联的协作环境中 [25], 这种协作环境会对其中的开发者群体之间的协作产生潜移默化的影响.

• 群体协作机制模型. 群体协作机制及模型包括大规模群体的组织模式、激励机制以及协作模型 3 部分. 其中, 组织模式对项目内群体间协作提供行为规则和流程约束, 激励机制对大众群体的参与提供相应激励, 协作模型则为群体间高效协作提供可供参考的协同方案. 本文第 3 节对群体化开发方法的 3 大核心机理进行了详细阐述, 可针对任务特点构建以大众化协同、开放式共享与持续性评估为一体的群体协作模型, 实现群体间高效协同、透明分享与准确评估.

• 群体协作支撑技术. 在大规模参与的群体化协作背景下, 群体协作支撑技术主要包括以参与者为中心的大众化协同支撑技术、以资源为中心的开放式共享支撑技术和面向动态演化的持续性评估支撑技术.

4.1 群体协作基础环境

群体协作基础环境是大规模参与者共同协作进行软件开发的基础环境, 可以分为两个层面, 一个是面向整个社区项目的宏观协作环境, 这一协作环境不因开发者或者开源项目的不同而不同; 另外一个是项目或者开发者所处的局部协作环境, 这一协作环境会随着项目或者开发者的不同而变化.

• 宏观协作环境. 在开源发展过程中, 形成了以 Apache 为代表的精品式协作社区、以 SourceForge 为代表的集市式协作社区和以 GitHub 为代表的社交化协作社区等典型的宏观协作环境. Apache 社区瞄准精品项目, 具有严格的准入门槛, 且项目之间存在紧密关联 [26]; SourceForge 面向大众群体, 提供项目托管服务, 群体参与不受限, 项目之间相对独立 [27]; GitHub 则将社交机制纳入软件开发, 提供了一个更为社交透明的协作环境 [28]. 宏观协作环境会对参与群体规模以及群体间协作效率等产生较大影响, 正因为如此, 大量流行的开源项目如 Ruby on Rails, Hibernate, phpadmin 等, 纷纷从传统的开源托管平台如 SourceForge 上迁移至 GitHub 社区, Apache 社区也有 70% 的项目在 GitHub 上创建了镜像以吸引用户的关注.

• 局部协作环境. 在宏观协作环境之上, 开发者以及项目相互关联形成了项目生态、参与者生态等, 将相关项目或者参与者连接起来, 形成以项目或者参与者为中心的局部协作环境. 其中, 项目生态是指项目间由于复用、扩展等关联而形成的软件生态系统, 如 Apache 社区中的项目复用依赖生态 [26]; 参与者生态是指开发者之间由于社交联系而形成的生态系统, 如 GitHub 中开发者之间由于 Follow, Star 等关系形成的生态 [29]; 此外, 开发者之间、开发者与项目之间以及项目与项目之间也可构成紧密联系的生态, 如 ruby 项目中开发者、ruby 的 gem 包以及他们之间的关联形成的 ruby 生态系统 [30]. 局部协作环境会对参与群体的协作产生直接的影响.

4.2 群体协作机制模型

在群体化开发环境中, 群体协作机制模型主要包括组织模式、激励机制以及协作模型. 其中, 组织模式定义了群体间协作的一般方式, 群体激励机制实现对参与群体的有效激励, 协作模型则提供了可参考的群体协作方案.

• 组织模式. 互联网环境下的群体化软件开发, 既需要充分发挥每个参与者个体的主观能动性, 创造超越个体的群体智能, 也需要对参与大众进行一定的约束和引导, 从而保证全局发展符合预期方向. 强组织机制在传统工业生产中发挥了巨大作用, 能够保证生产环节的进度和质量, 但是强约束方式限制了个体主观能动性的发挥. 弱组织机制则在互联网环境下得到广泛应用, 能够有效激发大众参与的积极性和创造力, 但是松散的机制难以确保整体目标的推进和实现. 两者之间既有较大差异, 又具有很大的互补性, 具体的对比如表 1 所示.

开源开发逐渐形成了一种类似 “洋葱” 结构的 “小核心 – 大外围” 的群体参与组织结构 [31], 实现了强组织规则与弱组织机制的融合. 在这种结构下, 小核心通常是项目的创始人及少量的核心参与者, 通常具有层次结构清晰、职责划分明确、规则约束很强的特点. 大外围通常是大量的外围开发者、用户以及其他利益相关者, 具有参与边界开放、参与者数量众多、角色多样、组织松散的特点. 在强组织规则下, 小核心决策了项目生产的技术路线、方向和推进进度等; 在弱组织规则下, 大外围在小核心的引导下, 积极参与项目具体任务的解决以及创作创意的贡献等. 通过这种 “小核心 – 大外围” 的组织

表 1 强组织与弱组织

Table 1 The comparison between strong and loose organization

	Strong organization	Loose organization
Expected target	Very specified	Not specified
Evaluation method	The degree to match the overall expectation	The degree to exceed individual expectation
Restriction way	What can do	What cannot do
Restriction degree	Strong restriction on behaviors	Weak restriction on behaviors
Subjective initiative	Individual initiative is constrained	Individual initiative is high

结构, 实现了群体化强组织与弱组织的有机融合, 有效推动了项目的发展. Linux 操作系统内核以及外围生态圈是这种组织结构的典型代表.

随着开源模式的发展, 越来越多的商业公司在开源项目中发挥了越来越多的作用, 通过安排全职员工的方式主导开源项目的发展, 使得强组织模式得到增强. 如何进一步地平衡强组织与弱组织的关系, 充分发挥外围大众的创意和核心团队的效率, 是群体化开发中需要持续探索解决的问题.

• 激励机制. 开源和众包模式下的参与者通常是利用自己的业余时间志愿参与其中, 他们可以参与长期贡献也可能随时退出, 如何吸引大规模群体积极参与和持续贡献需要有效的激励机制.

开源软件开发中的激励机制以声誉机制为主. 在 Linux, Apache 等项目中, 设定了不同的参与者角色, 对于不同角色的参与者赋予不同级别的权限, 如提交缺陷、提交代码、合并代码、发布软件等, 激励参与者积极贡献从而获得更高等级的参与权和决策权 [32]; 在 StackOverflow 社区中, 构建了以积分和权限相结合的声誉机制, 参与贡献越多、问答质量越高的用户积分越高, 同时具有的权限越高 [33]; 在众包协作中的激励机制以物质奖励为主. 在 Amazon 的 Mechanical Turk 平台, 任务发布者给每个任务设定相应的标价, 完成任务的参与者能够获得相应的奖金 [10]. 在激励机制的设计中, 物质奖励的额度设定会对参与者的多少以及任务完成质量产生很大的影响 [34].

在当前的开源模式中, 大规模的志愿参与者并未能够从参与开源获得等价的物质回报, 这也成为大量开发者未能成为项目长期贡献者的一个重要影响因素. 设计合理的知识产权共享机制, 让所有参与开源的志愿者共同享有参与开源项目劳动付出等价的知识产权, 并在开源项目发展成熟而产生物质收益时共享知识产权匹配的物质回报, 一方面将能够有效激励参与者持续参与贡献, 另一方面也可激励参与者贡献高质量的代码, 共同驱动项目高质量快速发展.

• 协作模型. 在互联网大规模群体自由参与下, 基于强组织与弱组织相结合的 “小核心 – 大外围” 组织模式实现了大规模外围群体与小规模核心团队之间的联接, 并将外围群体的创作活动融入到核心团队的生产过程. 但是, 外围群体如何高效参与, 核心团队如何有效评估, 从而在保证代码质量的同时提高协作效率, 是群体协作中需要解决的核心问题.

代码开发是开源群体协作的核心内容, 大规模开源群体间的协作模型随着代码版本管理工具的发展不断发展 [35]. 传统的代码版本通常采用集中式版本控制系统 (centralized version control system, CVCS) 进行管理, SVN (subversion) 是代表性的集中式版本管理工具. 基于集中式代码版本工具, 开源参与群体通常采用单点协作模型, 每个项目只有一个中心仓库, 每一个参与者直接与该中心仓库交互. 当某一个参与者修改并提交了代码, 那么其他人只能基于该参与者修改后的代码进行合并修改和提交. 基于 SVN 等代码管理工具的单点协作模型不利于大规模群体间的高效协作.

Git 分布式版本管理工具的出现有效改善了这种状况, 其具有的分布式特性使得开发者群体之间的协作方式变得灵活多样. 基于 Git 的群体协作, 通常每个项目有一个官方仓库, 同时每个参与者可

以基于该仓库复刻 (fork) 一个属于自己的独立仓库, 并将自己的开发活动暂时集中在个人仓库中. 当个人开发达到一定程度可以贡献给官方仓库时, 可以给官方仓库发送合并请求 (pull-request), 由官方仓库管理员进行审查后合并. 基于该模式, 参与者之间的开发活动可以在开发过程中相互隔离, 贡献汇聚时才进行合并处理, 极大地提高了大规模群体间的协作效率.

4.3 群体协作支撑技术

群体化开发的协作基础环境、协作机制模型的实现离不开相应技术的支撑. 为了提高群体协作效率, 学术界和工业界对此开展了广泛研究, 本小节围绕协同、共享与评估分别介绍相应的支撑技术.

(1) 大众化协同支撑技术. 大规模群体协作广泛存在于自然界, 如蚁群觅食、蜂群筑巢等生物活动, 并形成了以环境为媒介的基于环境激发效应的昆虫群体间间接协同. 在群体化软件开发过程中, 逐渐形成了以任务为媒介的群体间间接协同以及基于社交机制的群体间直接协同等典型研究技术和实践工具.

• 以任务为媒介的群体协同技术. 在软件项目层面, 参与者之间通过项目规划、任务指派等方式实现以任务为媒介的群体协同. Yu 等 [36] 和 Kim 等 [37] 研究了缺陷严重性预测方法, 对开发任务优先级进行规划; Naguib 等 [38] 和 Xuan 等 [39] 则提出了任务分派方法等. 在 GitHub 等开源社区, 项目管理者利用 Milestone 机制对阶段性任务进行规划 [40], 基于 pull-request、持续集成等机制实现子任务结果的汇聚. 在开发任务层面, 开发者之间通过版本控制工具实现细粒度的开发过程协同 [41]; 利用邮件列表、缺陷管理等工具实现针对具体任务的交互沟通 [42] 等.

• 基于社交机制的群体协同技术. 社交网络技术的发展对软件开发的群体协同产生了很大的影响, 以 GitHub 为代表的开源开发社区将 Follow, @ 等社交机制融入协作开发过程. Yu 等 [43] 研究了 GitHub 中参与者之间的 Follow 关系网络, 发现群体间存在团组、星型等多种不同的协同模式; Zhang 等 [44] 则分析发现 @ 机制的应用有效降低了 pull-request 的处理延迟. 此外, watch 机制、Star 机制等能够帮助参与者建立与其他开源项目和参与者等之间的直接联系 [45], 实现群体间的直接协同, 提高了大规模开发者群体间的协同效率.

(2) 开放式共享支撑技术. 互联网已成为开放的共享资源库, 大规模资源的结构化组织管理、扁平化联接、检索与推荐等已成为网络环境下资源聚合、管理与获取的重要手段.

• 基于标注的大规模资源管理技术. 给共享资源标注类别或者社会化标签是互联网环境下大规模资源有效组织管理的重要方法. 在学术界, 研究人员 [46~48] 提出了基于支持向量机、潜在语义索引、概率主题模型等技术的软件自动分类方法, Xia 等 [49] 和 Wang 等 [50] 提出了标签自动标注方法, 实现大规模软件资源的组织. 在工业界, SourceForge 社区定义了层次化的类别体系, 由项目管理者提交项目时选择所在类别, 实现对社区内项目的层次化组织; OpenHub, StackOverflow 等社区则设计了社会化标签机制, 允许注册用户给项目添加标签, 从而借助群体标注实现对大规模资源的组织.

• 基于联接的大规模资源聚合技术. 基于联接的方式能够将高度分散于不同社区的碎片化资源有效连接起来, 实现信息、资源的高效汇聚和共享 [51]. Correa 等 [52] 通过调查问卷和设计实验定量和定性地分析了跨社区链接对加快资源共享的重要作用, Wang 等 [53] 提出了跨社区缺陷与问答讨论自动连接的方法, Bacchelli 和其他一些研究者 [54,55] 则提出了基于信息检索、文本分析等不同的方法将邮件讨论与代码实现、API 与示例代码、API 文档与互联网资源以及源代码实体关联起来的方法, 实现关联资源的有效聚合, 加速资源的共享.

• 基于检索与推荐的大规模资源获取技术. 开放共享资源的大规模性使得群体协作中所需资源的准确定位面临巨大挑战, 研究学者在资源检索与推荐方面进行了深入研究. Yin 等 [24] 和 Bajracharya

等 [56] 通过软件源代码、软件描述的分析以及其他搜索引擎的聚合, 构建了相应的软件代码相关资源的检索系统, 文献 [57~59] 等则利用上下文信息以及文本分析等技术构建面向软件开发的互联网知识资源的检索与推荐系统.

(3) 持续性评估支撑技术. 持续性评估是群体协作有效推进的重要保障, 群体协作中的群体参与者、协作任务以及共享资源, 都需要相应持续评估技术的支撑.

• 面向参与者的持续评估技术. 群体协同中参与者的技术能力、兴趣关注点等方面会对协作效率和质量产生很大的影响, 学术界和工业界提出了一系列的相关技术来进行度量和评估. Nguyen 等 [60] 和 Canfora 等 [61] 通过参与者在开源协同开发社区中的历史开发行为和社交行为分析评估开发者的技术能力, Zhou 等 [62] 则提出了基于开发行为的能力量化方法, 对参与者随项目发展的能力成熟度进行持续评估. 开发者在知识分享社区中的参与活动从另一个角度反映了开发者的能力, 文献 [63~65] 等通过分析参与者在知识社区交流讨论行为实现对其技术水平和潜在能力的评估和预测.

• 面向协作任务的持续评估技术. 开发任务的准确评估直接影响到项目的开发质量和开发进展, 缺陷修复是软件开发过程中的重要开发任务, 缺陷优先级的准确预测是推动项目快速发展的重要环节. Hooimeijer 等 [66] 基于大规模缺陷样本利用统计分析方法设计了缺陷报告质量分析模型, Kim 等 [37] 和 Lamkanfi 等 [67] 分别采用分类算法、神经网络、网络中心度等方法对软件缺陷的优先级进行预测和评估, Yu 等 [68] 研究了持续集成技术在协作任务评估中的作用. 在众包实践中, 则主要通过专家评估、多人对比等方式实现对任务完成情况的评估 [69].

• 面向共享资源的持续评估技术. 互联网环境下共享资源存在质量参差不齐等问题, 对共享资源的准确评估对群体协作至关重要. 为给群体协作提供高质量的可复用开源资源, Bauer 等 [70] 和 Lavazza 等 [71] 通过静态代码分析方法对开源软件质量进行评估. Rudzki 等 [72] 则设计实现了一个包括活跃度、代码审阅、文档支持、软件社区和成熟度等多个维度的开源软件质量评估框架. Zou 等 [73] 则从用户角度入手提出了一种基于互联网用户评论的软件质量度量方法. 在知识分享社区通过构建完善的评估机制对其中的资源进行有效的评估. 例如, 在 StackOverflow 中采用用户点赞、评分的机制, 通过聚合大量阅读者的评分实现对资源的持续评价 [33].

5 Trustie 的群体化开发实践

开源软件的开发过程是一个 “大众参与创作” 的群体协作过程, 国防科技大学的 Trustie 团队以分布式版本控制工具 Git 为基础, 将协同、共享及评估机制融入其中, 构建形成了群体化开发支撑平台 Trustie [1)], 支持 “小核心” 与 “大外围” 的联接、“软件生产” 与 “软件创作” 的联接, 实现 “软件作品” 与 “软件产品” 的转换, 并围绕软件开发开展大规模的实践.

开源群体化开发过程具体如图 6 所示. 围绕开发任务, Trustie 平台建立了以任务为中心的大众化协作机制, 通过连接和集成任务跟踪管理、里程碑规划以及甘特图等工具, 围绕不同软件开发阶段形成外围大众与核心团队的连接和协作、创意需求汇聚与产品开发规划的衔接和转换. 核心开发团队依托 Trustie 平台在互联网上发布项目初始版本, 激发大众化群体体验, 基于任务跟踪管理模块发布使用反馈和需求创意, 此时项目处于发散的群体创作阶段; 核心团队分析外围群体的需求和创意, 进行有效性和优先级的评估和分类 [74], 并将其纳入到软件开发里程碑中, 并指派合适的开发者来完成相应的任务如 pull-request 审阅等 [75], 围绕这些任务的开发进入工程化开发的生产阶段. 围绕开发任务, 核心团队和外围大众可以持续交互讨论, 形成微社区.

1) https://www.trustie.net.

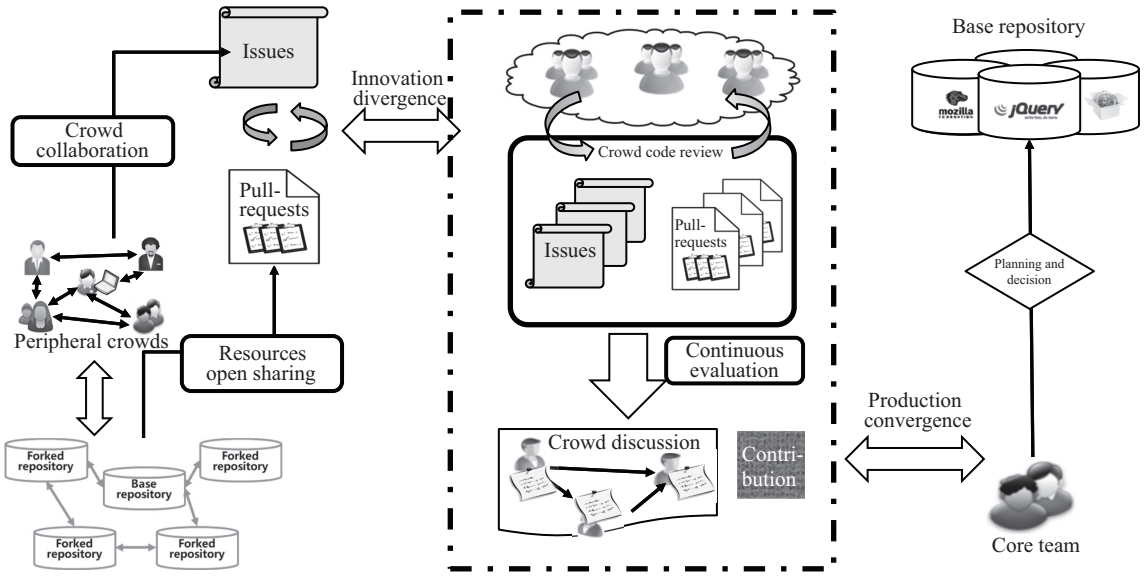


图 6 Trustie 中的群体化软件开发

Figure 6 The crowd software development process in Trustie

围绕开源项目, Trustie 平台形成了组织内部私有资源以及互联网社区开放资源相结合的开放式共享机制. 项目内部参与者可以通过平台的资源模块发布和共享设计文档、参考资料等不同类型的资源. 同时, Trustie 平台集成了面向全球开源社区的资源汇聚评估与检索排行平台 OSSESN 实现对可复用资源的检索和查找. OSSEAN 平台建立了覆盖 GitHub, SourceForge 等开发社区和 StackOverflow, OSChina 等应用社区的开源数据实时采集与关联分析机制, 基于社区活跃度、项目健康度、社区影响力等维度对开源资源进行动态、综合的评估和排行 [24], 为 Trustie 平台提供开放式资源共享服务.

围绕项目开发, Trustie 平台围绕开发者、开发资源以及群体贡献进行持续性评估. 例如, 核心团队针对外围大众的代码贡献的质量分析和评估, 基于对开发者能力分析进行 pull-request 审阅人推荐 [76], 基于对开源资源多维度评估的可复用资源检索与推荐 [24]. 依托持续性分析评估机制和技术, 实现群智高质量汇聚与收敛.

从群体的协作组织看, 开源项目形成了一种核心开发团队主导、大众化群体参与的协作组织. 核心开发团队是项目主导者, 在社区发布项目初始版本, 分析大众反馈, 评估大众贡献, 决策项目进度与发展方向; 大众化群体积极参与, 以多种形式, 如反馈使用体验、提出个体需求、提交贡献代码、参与审阅流程等进行贡献. 从项目的发展演化看, 针对不同的开发任务, 会经历创意汇聚和生产规划的不同阶段, 这是微观视角下具体任务的群体化开发过程; 从宏观视角看, 整个项目的创意激发汇聚和任务规划生产是交织在一起的, 不断推动项目持续迭代演化.

Trustie 平台和技术已在我国软件企业、人才培养和开源社区建设中得到大规模应用实践. 东软利用 Trustie 跨团队协作和开放式共享技术实现内部生产平台的改进升级, 在超过 20 个大型软件项目中的应用实验表明, 群体化方法极大提高了这些项目中的代码复用率、协作效率以及软件质量; Trustie 平台作为创新实践支撑平台在全国超过 650 家高校中推广应用, 支撑软件工程教育和信息技术实践教学改革, 有效提升了人才培养质量. 此外, Trustie 核心技术有效支持了我国包括 OpenI 启智开源平

台[2)]、GCC 绿色计算社区[3)] 等围绕人工智能、大数据等开源社区的建设, 为国内开源生态建设提供有力支撑.

6 未来展望

开源和众包的迅速发展和广泛实践展示了基于互联网的群智软件开发所蕴含的巨大创新潜力. 但是, 开源和众包仍然是群智软件开发的初始形态, 如何高效激发和稳态汇聚大规模群体智能, 确保群体智能在软件开发活动中可控形成和重复出现, 构建持续健康演化的软件生态, 是群智软件开发未来需要解决的核心挑战, 具体主要包括以下 3 方面.

- 参与群体的长效激发. 基于人类群体智能的软件开发不仅仅是一种技术问题, 更是一个心理、社会、经济等多种属性交织作用的复杂问题, 如何有效激发每一个参与个体进行持续高质量的贡献是群智开发需要解决的一个重大挑战.
- 群智贡献的高效融合. 开放参与下的软件开发具有群体贡献碎片化、群智结果不可预期性等特点, 如何量化度量群智贡献的质量和价值, 形成多源碎片化群智贡献的高效汇聚和快速收敛, 实现高效群智涌现是需要解决的关键挑战.
- 群智生态的持续演化. 群智生态中各个要素相互依赖、紧密交互, 如何建立多元高效的主动反馈机制, 对参与群体进行实时反馈和持续引导, 驱动群智生态的正向演化是群智软件开发需要解决的另一挑战.

本文在探究开源众包实践背后所蕴含的群体协作核心机理的基础上, 突破传统自动化和工业化方法的局限, 凝练了以大众化协同、开放式共享和持续性评估为核心要素的基于群智的软件开发群体化方法, 提出了以协同、共享和评估技术为支撑, 融合群体协作基础环境、协作机制模型的群体化开发环境, 实现创作与生产、创新与创收的融合与平衡, 为互联网环境下的软件开发提供重要的支撑.

参考文献

1 Yang F Q, Lü J, Mei H. Technical framework for Internetware: an architecture centric approach. Sci China Ser F-Inf Sci, 2008, 51: 610–622
2 Lehman M M, Belady L A. Program Evolution: Processes of Software Change. London: Academic Press Professional, Inc., 1985
3 Li D Y. Software engineering in Internet age. China Comput Federation Lett, 2009, 35: 7–12 [李德毅. 网络时代的软件工程. 中国计算机学会通讯, 2009, 35: 7–12]
4 Raymond E. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. Sebastopol: O'Reilly Media, Inc., 2001
5 Howe J. The rise of crowdsourcing. Wired Magaz, 2006, 14: 1–4
6 Zhang W, Mei H. Software development based on collective intelligence on the Internet: feasibility, state-of-the-practice, and challenges. Sci Sin Inform, 2017, 47: 1601–1622 [张伟, 梅宏. 基于互联网群体智能的软件开发: 可行性、现状与挑战. 中国科学: 信息科学, 2017, 47: 1601–1622]
7 Lerner J, Tirole J. Some simple economics of open source. J Ind Econ, 2000, 50: 197–234
8 DiBona C, Cooper D, Stone M. Open Sources 2.0: The Continuing Evolution. Sebastopol: O'Reilly Media, Inc., 2005
9 Brabham D C. Crowdsourcing as a model for problem solving: an introduction and cases. Int J Res New Media Tech, 2008, 14: 75–90

2) https://openi.org.cn/.
3) http://opengcc.org/.

10 Heer J, Bostock M. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2010. 203–212
11 Hagel J, Brown S J. The next wave of open innovation: how innocentive aims to exploit sophisticated technology and networking capabilities to connect problems with their potential solvers. Business Week, 2009
12 Lakhani K R, Garvin D A, Lonstein E. Topcoder (A): developing software through crowdsourcing. Harvard Business School General Management Unit Case No. 610-032. 2010.
13 Howe J. Crowdsourcing: why the power of the crowd is driving the future of business. Am J Health-Syst Pharmacy, 2010, 67: 1565–1566
14 Zhou M, Mockus A, Ma X, et al. Inflow and retention in OSS communities with commercial involvement: a case study of three hybrid projects. ACM Trans Softw Eng Methodol, 2016, 25: 1–29
15 Chawla S, Hartline J D, Sivan B. Optimal crowdsourcing contests. Games Economic Behav, 2019, 113: 80–96
16 Li W, Wu W, Wang H, et al. Crowd intelligence in AI 2.0 era. Front Inf Techn Electron Eng, 2017, 18: 15–43
17 Strübing J. Designing the working process — what programmers do beside programming. In: User-Centred Requirements for Software Engineering Environments. Berlin: Springer, 1994. 81–90
18 Michelucci P, Dickinson J L. The power of crowds. Science, 2016, 351: 32–33
19 Wang H M, Yin G, Xie B, et al. Research on network-based large-scale collaborative development and evolution of trustworthy software. Sci Sin Inform, 2014, 44: 1–19 [王怀民, 尹刚, 谢冰, 等. 基于网络的可信软件大规模协同开发与演化. 中国科学: 信息科学, 2014, 44: 1–19]
20 Wang H M, Wu W J, Mao X J, et al. Growing construction and adaptive evolution of complex software systems. Sci Sin Inform, 2014, 44: 743–761 [王怀民, 吴文峻, 毛新军, 等. 复杂软件系统的成长性构造与适应性演化. 中国科学: 信息科学, 2014, 44: 743–761]
21 Zhou M, Mockus A. Growth of newcomer competence: challenges of globalization. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, 2010. 443–448
22 Wang H, Yin G, Li X, et al. TRUSTIE: a software development platform for crowdsourcing. In: Crowdsourcing. Berlin: Springer, 2015. 165–190
23 Rudzki J, Kiviluoma K, Poikonen T, et al. Evaluating quality of open source components for reuse-intensive commercial solutions. In: Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009
24 Yin G, Wang T, Wang H M, et al. Ossean: mining crowd wisdom in open source communities. In: Proceedings of IEEE Symposium on Service-Oriented System Engineering, 2015. 367–371
25 Zhou M, Mockus A. Does the initial environment impact the future of developers? In: Proceedings of the 33rd International Conference on Software Engineering, 2011. 271–280
26 Bavota G, Canfora G, Penta M D, et al. The evolution of project inter-dependencies in a software ecosystem: the case of apache. In: Proceedings of IEEE International Conference on Software Maintenance, 2013. 280–289
27 Gonzalez-Barahona J M, Robles G, Andradas-Izquierdo R, et al. Geographic origin of libre software developers. Inf Economics Policy, 2008, 20: 356–363
28 Dabbish L A, Stuart H C, Tsay J, et al. Social coding in GitHub: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, Seattle, 2012. 1277–1286GitHub
29 Wu Y, Kropczynski J, Shih P, et al. Exploring the ecosystem of software developers on GitHub and other platforms. In: Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, 2014. 265–268
30 Kabbedijk J, Jansen S. Steering insight: an exploration of the ruby software ecosystem. In: Proceedings of International Conference of Software Business, Brussels, 2011. 44–55
31 Nakakoji K, Yamamoto Y, Nishinaka Y, et al. Evolution patterns of open-source software systems and communities. In: Proceedings of International Workshop on Principles of Software Evolution, 2002. 76–85
32 Fielding R T. Shared leadership in the apache project. Commun Acm, 1999, 42: 42–43
33 Mamykina L, Manoim B, Mittal M, et al. Design lessons from the fastest Q&A site in the west. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2011. 2857–2866
34 Mason W, Watts D J. Financial incentives and the “performance of crowds”. SIGKDD Explor Newsl, 2010, 11:

100–108
35 Brindescu C, Codoban M, Shmarkatiuk S, et al. How do centralized and distributed version control systems impact software changes? In: Proceedings of the 36th International Conference on Software Engineering, 2014. 322–333
36 Yu L, Tsai W-T, Zhao W, et al. Predicting defect priority based on neural networks. In: Proceedings of the 6th International Conference on Advanced Data Mining and Applications, Chongqing, 2010
37 Kim S, Ernst M D. Which warnings should I fix first? In: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2007. 45–54
38 Naguib H, Narayan N, Bruegge B, et al. Bug report assignee recommendation using activity profiles. In: Proceedings of the 10th Working Conference on Mining Software Repositories, 2013. 22–30
39 Xuan J, Jiang H, Ren Z, et al. Developer prioritization in bug repositories. In: Proceedings of the 34th International Conference on Software Engineering (ICSE), 2012. 25–35
40 Salo R. A guideline for requirements management in GitHub with lean approach. Dissertation for Master's Degree. Tampere: University of Tampere, 2014
41 Brindescu C, Codoban M, Shmarkatiuk S, et al. How do centralized and distributed version control systems impact software changes? In: Proceedings of the 36th International Conference on Software Engineering, 2014. 322–333
42 Serrano N, Ciordia I. Bugzilla, itracker, and other bug trackers. IEEE Softw, 2005, 22: 11–13
43 Yu Y, Yin G, Wang H M, et al. Exploring the patterns of social behavior in GitHub. In: Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies, 2014. 31–36
44 Zhang Y, Wang H, Yin G, et al. Exploring the use of @-mention to assist software development in GitHub. In: Proceedings of the 7th Asia-Pacific Symposium on Internetware, 2015. 83–92
45 Lee M J, Ferwerda B, Choi J, et al. GitHub developers use rockstars to overcome overflow of news. In: Proceedings of Extended Abstracts on Human Factors in Computing Systems, 2013. 133–138
46 Mcmillan C, Linaresvasquez M, Poshyvanyk D, et al. Categorizing software applications for maintenance. In: Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM), 2011. 343–352
47 Tian K, Revelle M, Poshyvanyk D. Using latent dirichlet allocation for automatic categorization of software. In: Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories, 2009. 163–166
48 Wang T, Wang H, Yin G, et al. Mining software profile across multiple repositories for hierarchical categorization. In: Proceedings of IEEE International Conference on Software Maintenance, 2013. 240–249
49 Xia X, Lo D, Wang X, et al. Tag recommendation in software information sites. In: Proceedings of the 10th Working Conference on Mining Software Repositories, 2013. 287–296
50 Wang T, Wang H, Yin G, et al. Tag recommendation for open source software. Front Comput Sci, 2014, 8: 69–82
51 Gomez C, Cleary B, Singer L. A study of innovation diffusion through link sharing on stack overflow. In: Proceedings of the 10th Working Conference on Mining Software Repositories, 2013. 81–84
52 Correa D, Lal S, Saini A, et al. Samekana: a browser extension for including relevant web links in issue tracking system discussion forum. In: Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC), 2013. 25–33
53 Wang H, Wang T, Yin G, et al. Linking issue tracker with Q&A sites for knowledge sharing across communities. IEEE Trans Serv Comput, 2018, 11: 782–795
54 Bacchelli A, Lanza M, Robbes R. Linking e-mails and source code artifacts. In: Proceedings of 2010 ACM/IEEE 32nd International Conference on Software Engineering, 2010. 375–384
55 Subramanian S, Inozemtseva L, Holmes R. Live API documentation. In: Proceedings of the 36th International Conference on Software Engineering, 2014. 643–652
56 Bajracharya S, Ossher J, Lopes C. Sourcerer: an internet-scale software repository. In: Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation, 2009. 1–4
57 Gottipati S, Lo D, Jiang J. Finding relevant answers in software forums. In: Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, 2011. 323–332
58 Brandt J, Dontcheva M, Weskamp M, et al. Example-centric programming: integrating web search into the development environment. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2010. 513–522
59 Bacchelli A, Ponzanelli L, Lanza M. Harnessing stack overflow for the IDE. In: Proceedings of the 3rd International

Workshop on Recommendation Systems for Software Engineering, 2012. 26–30
60 Nguyen T T, Nguyen T N, Duesterwald E, et al. Inferring developer expertise through defect analysis. In: Proceedings of the 34th International Conference on Software Engineering, 2012. 1297–1300
61 Canfora G, Penta M D, Oliveto R, et al. Who is going to mentor newcomers in open source projects? In: Proceedings of the 20th ACM SIGSOFT International Symposium on the Foundations of Software Engineering, 2012. 44
62 Zhou M, Mockus A. Developer fluency: achieving true mastery in software projects. In: Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010. 137–146
63 Sinha V S, Mani S, Gupta M. Exploring activeness of users in Q&A forums. In: Proceedings of 2013 10th Working Conference on Mining Software Repositories, 2013. 77–80
64 Pal A, Harper F M, Konstan J A. Exploring question selection bias to identify experts and potential experts in community question answering. ACM Trans Inf Syst, 2012, 30: 10
65 Surian D, Liu N, Lo D, et al. Recommending people in developers' collaboration network. In: Proceedings of the 18th Working Conference on Reverse Engineering, 2011. 379–388
66 Hooimeijer P, Weimer W. Modeling bug report quality. In: Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, 2007. 34–43
67 Lamkanfi A, Demeyer S, Giger E. Predicting the severity of a reported bug. In: Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), 2010. 1–10
68 Vasilescu B, Yu Y, Wang H, et al. Quality and productivity outcomes relating to continuous integration in GitHub. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015. 805–816
69 Snow R, Oconnor B, Jurafsky D, et al. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2008. 254–263
70 Bauer V, Heinemann L, Hummel B, et al. A framework for incremental quality analysis of large software systems. In: Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM), 2012. 537–546
71 Lavazza L, Morasca S, Taibi D, et al. Predicting oss trustworthiness on the basis of elementary code assessment. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2010. 1–4
72 Rudzki J, Kiviluoma K, Poikonen T, et al. Evaluating quality of open source components for reuse-intensive commercial solutions. In: Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009. 11–19
73 Zou Y, Liu C, Jin Y, et al. Assessing software quality through web comment search and analysis. In: Proceedings of International Conference on Software Reuse, 2013. 208–223
74 Fan Q, Yu Y, Yin G, et al. Where is the road for issue reports classification based on text mining? In: Proceedings of 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2017. 121–130
75 Yu Y, Wang H, Yin G, et al. Reviewer recommendation for pull-requests in GitHub. Inf Softw Tech, 2016, 74: 204–218
76 Wang H. Harnessing the crowd wisdom for software trustworthiness. ACM Sigsoft Softw Eng Notes, 2018, 43: 1–6

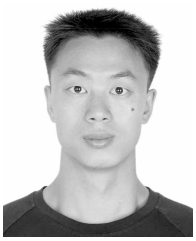
Crowd-intelligence-based software development method and practices

Tao WANG[1,2*], Gang YIN[2], Yue YU[1,2], Yang ZHANG[1,2] & Huaimin WANG[1,2*]

1. *National Laboratory for Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha* 410073*, China*;
2. *Laboratory of Software Engineering for Complex Systems, Changsha* 410073*, China*
* Corresponding author. E-mail: taowang2005@nudt.edu.cn, whm_w@163.com

Abstract The evolution of Internet has produced profound impacts on software development technologies, operation forms, and service models. In particular, the underlying crowd intelligence mechanisms in large-scale crowd collaboration practices, such as open sourcing and crowdsourcing, have greatly inspired the software development. Thus, this study investigates the innovation and production models for such open-sourcing and crowdsourcing practices. Based on this analysis, we propose the crowd-intelligence-based software development mechanisms including crowd collaboration, open sharing, and continuous evaluation. Specifically, we study the basic environment, the mechanism model, and the key technologies of crowd collaboration. Based on these aspects, we comprehensively discuss the crowd-intelligence-based software development environment, the involved core factors, and our open-sourcing practice. Finally, we propose the great challenges to be addressed in the future, and hope these studies and discussions are helpful.

Keywords open source, crowdsourcing, crowd intelligence, crowd development, crowd collaboration, open sharing, continuous evaluation



Tao WANG was born in 1984. He received the Ph.D. degree in computer science from National University of Defense Technology, ChangSha, in 2014. Currently, he is an association professor at National University of Defense Technology. His research interests include software repository mining, crowd-intelligence-based software engineering, and open source ecosystem.

Gang YIN was born in 1975. He received the Ph.D. degree in computer science from National University of Defense Technology, Changsha, in 2006. Currently, he is an association professor at National University of Defense Technology. His research interests include software engineering, software education, and open source ecosystem.

Yue YU was born in 1988. He received the Ph.D. degree in software engineering from National University of Defense Technology, Changsha, in 2016. Currently, he is an association professor at National University of Defense Technology. His research interests include software engineering, social coding, and open source ecosystem.

Huaimin WANG was born in 1962. He received the Ph.D. degree in software engineering from National University of Defense Technology, Changsha, in 1992. He is currently a professor at National University of Defense Technology. He is a member of the Chinese Academy of Sciences. His research interests include trustworthy computing, crowd intelligence, and distributed computing.