基于互联网群体智能的软件开发: 可行性、现状与挑战

张伟^{1,2*}, 梅宏^{1,2,3}

1. 高可信软件技术教育部重点实验室 (北京大学), 北京 100871

2. 北京大学信息科学技术学院软件研究所, 北京 100871

3. 北京理工大学计算机学院, 北京 100081

* 通信作者. E-mail: zhangw.sei@pku.edu.cn

收稿日期: 2017-05-20; 接受日期: 2017-07-21; 网络出版日期: 2017-11-28

国家自然科学基金 (批准号: 61690200, 61432020) 和国家重点基础研究发展计划 (973) (批准号: 2015CB352201) 资助项目

摘要 我们正在经历一场软件开发方式的变革. 在这场变革中, 互联网承担了双重角色: 一方面, 互联网形成了一个开放、动态、持续演化的软件运行环境, 极大地增加了软件系统的规模和复杂性, 为软件开发带来了新的挑战; 另一方面, 在“将计算机互联”和“将物品互联”两种基础角色之上, 互联网正在展现出“将智能互联”的崭新角色, 促成了群体智能现象在各种问题领域中的出现和快速发展, 为解决软件开发面临的本质性困难提供了一种潜在的创新性解决方案. 本文从可行性、实践现状、问题与挑战等几个方面对基于互联网群体智能的软件开发进行了概要性的论述和分析, 并对其理想形态进行了初步的探索. 我们希望通过这些论述和分析, 能够为软件工程领域的研究者对相关问题及其研究实践现状的理解提供一种有价值的观察视角, 为软件工程在互联网环境下的创新性发展贡献一些可能有用的信息和观点.

关键词 软件工程, 互联网, 群体智能, 开源软件, 软件众包

1 引言

1.1 关于“软件开发没有银弹”的再思考

1987年, 软件工程领域的著名学者 Frederick Brooks 在其经典论文 “No silver bullet: essence and accidents of software engineering”^[1] 中作出如下断言: “在未来 10 年内, 不会出现一种单一的技术或管理方法, 使得软件的生产效率、可靠性或易理解性能够得到数量级的提升”. 该断言所基于的论据是: 在当时已经存在或未来十年内可预见的技术和方法中, 不存在一种能够有效应对软件开发本质性困难的技术或管理方法. 大约 10 年后, 在 1995 年, Brooks 认为 “软件开发没有银弹” 的断言在下一个 10 年

引用格式: 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, doi: 10.1360/N112017-00117

Zhang W, Mei H. Software development based on collective intelligence on the Internet: feasibility, state-of-the-practice, and challenges (in Chinese). Sci Sin Inform, 2017, 47: 1601–1622, doi: 10.1360/N112017-00117

中仍然成立^[2]。现在是 2017 年, 距离 Brooks 第二次断言有效期的终止时间 (2005 年) 又过去了 12 年。一个自然出现的问题是: 软件开发没有银弹的断言是否仍然有效? 如果仍然有效, 是否会在将来的某个时间不再有效?

在一般意义上, 我们应该用发展的视角去看待任何问题, 即, 根据问题当前所处的上下文 (context), 寻求解决问题的方法。在一种上下文中无解的问题, 在另一种上下文中则可能是有解的。对于软件开发而言, 自 1987 年以来, 很多事情并没有发生显著的变化: Brooks 所指出的软件开发的本质性困难仍然存在 (虽然有所缓解)。但还有很多事情已经发生了显著的变化: 目前, 软件开发者和用户群体的规模得到了巨大的增长; 软件开发者具有了更高的个体生产效率, 而且能够通过更加方便的手段进行开发者个体之间的协同; 软件用户对软件也更加熟悉, 而且能够使用更加具体和明确的方式来表达他们的需求。因此, 从发展的视角来看, 一个更加具体的问题是: 在这样一种发生了显著变化的上下文中, 是否有可能发现一种有效应对软件开发本质性困难的途径?

不可否认, Brooks 关于软件开发没有银弹的断言蕴含了深刻的洞察力。但同时, 我们也要注意, 这个断言具有两个隐含的假设。第 1 个假设是关于软件开发中开发者的组织方式。Brooks 的软件开发经验主要来自于由其作为负责人的 IBM OS/360 操作系统软件开发项目。在这个项目中获得的软件开发经验, 促使其形成了《人月神话》^[2,3] 这一软件工程领域的经典著作。从这本书的描述中, 可以看到, Brooks 关于软件开发的经验主要建立在集中式层级结构 (centralized hierarchy)^[4] 这样一种组织方式之上。在这种组织方式中, 大部分的重要决策由层级结构中居于顶层的小部分领导者作出, 然后逐层向下传播。一方面, 集中式层级结构使得一个群体表现出高度的有序性; 另一方面, 集中式层级结构也牺牲了大部分低层个体具有的主观能动性、创造力和灵活性。如果采用其他类型的群体组织方式, 是否会对这个断言产生某些影响? 第 2 个假设是关于感受软件开发本质性困难的主体。很明显, 感受这种困难的主体是软件开发者、软件的利益相关者, 或者更一般地说, 是人类个体。但是, 如果让其他非人类的智能体来进行软件开发, 它们是否一定还会感受到这些本质性困难。可以想象, 一个被人类视为困难的问题, 对于一个智能水平远超过人类个体的智能体而言, 则有可能不再是一个困难问题。当然, 识别出这一隐含假设并不具有任何现实的意义, 除非我们能发现一种智能水平远超过人类个体的智能体; 或者更为激进, 除非我们能够通过某种方式来构造出一种智能水平远超过人类个体的智能体。

作为一名科研工作者, Brooks 具有非常严谨的思维逻辑: 他为自己的断言加上了一个十年的有效期, 因为他不认为人类能够对十年之后的事物作出有效的判断或预言。我们认为, Brooks “人类预言十年有效期” 这一观点是非常正确的。为了论证软件开发没有银弹这一断言, Brooks 按照自己的判断列出了在未来十年中所有可能流行的重要技术 (就当时而言), 其中包括: 面向对象的编程、自动化编程, 以及软件开发环境和工具等, 并逐一分析指出这些技术都无法解决软件开发的本质性困难。但是, Brooks 确实没有预见到十年之后出现的一项重要技术进展——互联网——对软件开发可能产生的影响。作为人类社会在 20 世纪最为重要的技术进展之一, 互联网对包括软件开发在内的大部分人类社会活动均产生了巨大而深远的影响。

1.2 基于互联网群体智能的软件开发

1.2.1 群体智能

长期以来, 科学家们在很多低等生物/动物群体中观察到一种看似矛盾的现象: 虽然构成群体的每一个体都不具有智能或只具有有限的智能, 但整个群体却表现出远超过任一个体能力的智能行为^[5~7]。科学家们将这种在低等生物/动物群体层次上展现出的智能现象, 称为“群体智能” (swarm

intelligence) [8]. 从哲学的角度观察, 群体智能是一种“由量变产生质变”的现象 [9]; 从复杂系统的角度观察, 群体智能是一种“涌现”现象, 是群体中的个体通过复杂的交互而涌现产生的一种“自组织行为” [10].

群体智能并不是低等生物/动物群体所特有的现象: 在人类个体和人类社会中, 都可以观察到群体智能现象 [11, 12]. 一方面, 从人脑的神经结构来看, 人类个体的智能本身即是一种群体智能现象: 这个群体中包含了 1000 亿左右的神经元个体 [13]; 人类个体具有的复杂认知和心理功能正是在这样一个大规模神经元群体及其构成个体之间复杂交互的基础上涌现产生的. 另一方面, 人类社会的不断发展和演化也可以被认为是一种群体智能现象: 这个群体中包含了特定人类社会中存在或曾经存在的所有人类个体; 人类社会的很多重要文明成果都是人类个体在长期群体化、社会化的生活中逐渐演化形成的产物; 离开了群体化、社会化的生活环境, 人类不可能形成目前的文明形态.

区别于面向低等昆虫和动物群体的“Swarm Intelligence”, 人们一般将面向人类群体的群体智能称为“Collective Intelligence”. 基于统一性的考虑, 将这两类智能均称为“群体智能”. 这两种群体智能的统一性体现在其对个体智能的增强或放大作用, 即, 无论个体具备何种级别的智能, 在群体层次上展现出的智能都会超过群体中任一个体的智能.

1.2.2 互联网群体智能

自然界中存在的群体智能现象 [5~8] 指出了一种对个体智能进行有效放大的可能性: 一群具有特定智能水平的个体, 通过大规模的群体协同, 可以在群体层次上展现出超过任一个体的智能水平. 自然界群体智能现象的发生存在两个基本条件: 一个大规模的群体; 一个公共的环境供群体在其中进行大规模的交互. 受限于物理时空的制约, 在特定的物理空间中, 在特定的时间段内, 进行人类群体的大规模持续密集的群体协同, 通常需要付出高昂的成本, 因而不具有一般意义上的可行性.

互联网的出现打破了物理时空对大规模人类群体协同的限制, 促成了基于互联网的人类群体的出现. 互联网重新定义了“两个人类个体之间的距离”这一概念: 任何一群地理分布的人类个体都有可能在互联网上形成一个具有紧密联系的群体. 同时, 互联网使得人类信息的总量、信息传播的速度和广度都在快速增长. 基于互联网的人类群体的出现为基于互联网的群体智能 (简称为“互联网群体智能”) 的形成提供了基本的前提条件. 互联网的不断发展则促成了互联网群体智能现象在各种问题领域内的不断涌现和蓬勃发展 [14], 其涉及的领域包括: 知识收集¹⁾、文本识别²⁾、产品设计³⁾⁴⁾、科学研究 [15] 以及软件开发 [16~18] 等. 虽然针对的领域和问题不同, 但这些互联网群体智能现象均体现出 3 个重要特点: 一个开放的大规模人类群体、群体中个体之间的直接或间接交互, 以及大量个体行为的汇聚或融合. 这些自发或通过自由探索形成的互联网群体智能现象表明了面向人类群体基于群体智能解决现实问题的可能性. 一个值得进一步思考的问题是: 群体智能现象背后是否存在一般性的基本原理, 使得我们能够利用这些基本原理设计出更加高效的人类群体智能系统, 从而为那些人类目前还无法解决或无法有效解决的困难问题探索新的解决方案. 例如, 能否基于群体智能对个体智能的放大作用, 有效应对软件开发对于人类个体而言所具有的本质性困难.

互联网群体智能现象的出现和蓬勃发展, 标志着在“将计算机互联” (internet-of-computers) 和“将物品互联” (internet-of-things) 这两种基础角色之上, 互联网正在逐步展现出“将智能互联” (internet-

1) 维基百科: Wikipedia. <http://en.wikipedia.org/wiki/Wikipedia>.

2) 基于软件登录验证码的文本识别: reCAPTCHA. <https://www.google.com/recaptcha>.

3) 衬衫设计: Theadless. <http://www.threadless.com>.

4) 积木设计: Lego Ideas. <https://ideas.lego.com>.

of-intelligences) 的崭新角色。

1.2.3 基于互联网群体智能的软件开发

软件开发是一种智力密集型活动, 而大规模复杂软件的开发则是一种面向群体的智力密集型活动。从发展历史来看, 软件的规模和复杂性呈现出不断增长的趋势。例如, Windows 操作系统的代码量从初期的 1000 万行 (Windows NT 4.0, 1996 年) 逐渐增加到 5000 万行 (Windows Vista, 2007 年)⁵⁾; 目前, 一辆高端轿车中的软件代码总量在 1 亿行左右⁶⁾。互联网的出现和不断发展则将软件的规模和复杂性提升到了一个崭新的层次: 互联网的连接属性使得不同的软件系统相互连接, 从而形成一种更加复杂的大规模软件生态系统。传统的软件开发方法已经无法有效应对互联网环境下软件规模和复杂性的显著提升。在“有效提高软件开发效率和质量”这一目标的驱动下, 实践者开始探索软件开发的新方式。

这些探索的一个重要方向即是基于互联网的群体化软件开发, 即, 如何利用互联网及相关技术在信息处理和传播效率方面的优势, 如何有效汇聚基于互联网的软件利益相关者群体中不同个体的能力, 提高软件开发的质量和效率。我们认为, 在这一探索方向上, 目前已经形成了 3 类重要的软件开发实践: 开源软件 (open source software)^[19]、软件众包 (software crowdsourcing)^{[16]7)}、应用程序商店 (application store)⁸⁾等。例如, TopCoder 通过众包方式, 吸引了分布于全球各地的软件开发者加入面向不同客户和问题领域的软件开发活动中; 利用群体的协作和竞争, 不仅提高了特定软件开发活动的速度, 还有效提高了产品的质量^[16]。越来越多的商业公司将其主要或核心业务建立在开源软件、软件众包以及应用程序商店之上, 在很大程度上验证了这些新型软件开发实践具有的巨大商业价值。但是, 上述几种基于互联网的群体化软件开发实践在开发者群体的规模、个体之间的交互, 以及个体行为的有效聚合方面, 与理想的互联网群体智能现象 (即, 一个大规模的互联网群体在完全去中心化的情况下采用自组织的方式对一个复杂问题进行求解) 相比^[20~25], 还存在较大的差距: 对于软件这样一种复杂的人工制品而言, 如何基于群体智能的方式对其进行构造和演化, 仍然存在巨大的研究和发展空间。同时, 互联网群体智能以及基于互联网的群体化软件开发正逐渐引起研究者的重视。1994 年, 法国哲学家 Pierre Lévy^[26] 从社会、经济以及政治的角度对信息空间 (cyberspace) 中人类群体智能的形成、特点及影响进行了系统性的分析与展望。1995 年以来, 研究者还提出了 Global Brain^[27~29] 和 Human Computation^[30,31] 等概念; 这些概念在目标上与互联网群体智能并没有本质性的差别。在软件开发领域中, 由实践者探索出的行之有效的基于互联网的群体化软件开发方式对软件研究领域产生了巨大的冲击, 吸引了一大批研究者对这些新型软件开发实践的各个方面进行经验性的解释和分析^[19,25,32~38]。而在传统软件开发方法的研究中, 受到各种因素的影响, 基于互联网实施特定软件开发活动的现象正逐渐萌芽^[39~41]。基于互联网的群体化软件开发在国内也开始受到研究者的重视^{[42~44]9)}。

我们认为, 现有的几种基于互联网的群体化软件开发实践, 仅仅是基于互联网群体智能的软件开发的一种原始形态。目前人们对于“基于互联网群体智能的软件开发”的认识和理解仍然处在必然王国的阶段: 虽然能够从目前基于互联网的群体化软件开发实践中观察到很多体现群体智能的实际案

5) <http://www.informationisbeautiful.net/visualizations/million-lines-of-code>.

6) <http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code>.

7) MATLAB 在线程序设计竞赛. <http://www.mathworks.com/matlabcentral/contest/>.

8) 苹果 iOS 应用程序商店. <https://itunes.apple.com/us/genre/ios/id36?mt=8>; 谷歌 Android 应用程序商店. <http://www.androidcentral.com/google-play-store>.

9) 群体软件工程专题. 中国计算机学会技术动态, 第 494 期, 2013 年 4 月 7 日。

例,但对于其背后的基本规律和关键机理仍然缺乏充分的认知.

对“基于互联网群体智能的软件开发”认知和理解的困难主要体现在 3 个方面. 第一,群体智能及互联网群体智能自身就是一种复杂的现象: 现有的研究还未能充分地揭示这种现象背后蕴含的规律和机理. 第二,软件以及软件开发活动自身的复杂性: 软件工程学科在几十年的发展过程中,从多种角度对如何控制软件及软件开发的复杂性进行了大量的探索,使得人们能够在一定范围内对软件及软件开发的复杂性进行有效地控制;但是,当软件的规模和复杂性超出了一定的范围,就会使得软件在开发效率、成本及质量上无法有效满足实际的需求;这种现象表明,现有研究和实践还不能在本质上对软件及软件开发的复杂性进行有效控制. 第三,对于将“互联网群体智能”这种一般性的机制应用到“软件开发”这种特定的问题领域所产生的特殊性,人们还缺乏基本的认知: 目前大量出现的互联网群体智能现象虽然在表象上体现出很强的共性特征,但是,不同问题领域的互联网群体智能现象,在具体机制和细节上却存在较大的差异;这种现象表明,在将互联网群体智能机制应用到特定问题领域的过程中,需要结合当前问题领域的特点,对这种一般性的机制进行针对性,甚至创新性的定制和扩展;对于软件这样一种具有严密逻辑结构的复杂知识制品,如果采用互联网群体智能的方式对其进行构造,其面临的特殊性是什么,如何基于这种特殊性对一般性的群体智能机制进行定制和扩展,目前的研究还缺乏对这些问题系统和深入的认识.

对于基本规律和关键机理认知的缺乏,导致人们无法采用一种科学和系统的方式使得“基于互联网群体智能的软件开发现象”可控地重复发生. 一方面,这说明“基于互联网群体智能的软件开发”目前还远远不是一种系统性的软件开发方法. 另一方面,这种现状也蕴含着重大的研究机遇: 从目前仍处于原始形态的“基于互联网群体智能的软件开发现象”中,我们看到了一种变革性、创新性、更高效的软件开发方式的可能性;如果这种软件开发方式能够成功地升级为一种系统性的软件开发方法,那么,将会为整个软件产业带来革命性的影响,为人类信息社会的文明形态奠定重要的基础设施^[26].

2 基于互联网群体智能的软件开发: 可行性分析

本节从 3 个方面对基于互联网群体智能的软件开发的可行性进行分析: 从软件开发本质性困难难以克服的主要原因出发,指出应对软件开发本质性困难的一种理论上可行的方式是求助于具有更高智能水平的智能体;从软件开发组织方式的演化趋势出发,指出我们可以求助的具有更高智能水平的智能体就蕴含在大规模的人类群体协同之中;从软件制品的结构出发,指出基于互联网群体智能进行软件制品的生产具有原理上的可行性.

2.1 软件开发本质性困难难以克服的主要原因

Brooks 指出,软件开发具有 4 种类型的本质性困难 (essential difficulties)^[1]: 复杂性、一致性、易变性、不可见性. 复杂性指的是软件涉及多方面的因素,且不同因素之间存在复杂的相互关系;复杂性要求在软件开发中能够全面地识别出这些多方面的因素及其之间的复杂关系. 一致性指的是软件不同因素之间存在众多隐含的一致性约束条件;一致性使得开发者很难在软件开发中保证这些隐含一致性约束条件的满足. 易变性指的是软件涉及的众多因素很容易发生变化;易变性要求软件也要随着相关因素的变化而发生相应的变化,而在软件不具有足够自适应能力的情况下,传统的软件开发方法无法保证软件的持续演化. 不可见性指的是软件没有视觉可见的物理形态;不可见性使得人类不再能够充分利用其视觉对软件的各个方面进行直观地观察,进而无法在此基础上对软件进行更深层次的理解、

图 1 关于人类群体组织方式的一个概念框架

Figure 1 A conceptual framework for organizational mechanisms of human collectives

分析和改进.

我们认为, 软件开发本质性困难难以克服的主要原因来自两个紧密相关的方面.

一方面, 软件设计和构造过程的开放性和灵活性, 极大地释放了人类个体智能具有的想象力和创造力, 使得人们可以在思维中想象或创造出远比物理制品更为复杂的抽象软件系统. 软件是一种没有物理形态的逻辑型人工制品. 在软件出现以前, 大部分的人工制品都具有物理形态. 物理制品所必然遵循的物理规律为物理制品的结构和制造过程施加了严格的限制, 使得人们可以采用相对固化或机械的方式对物理制品进行设计和生产. 相对于物理制品而言, 软件这种不具有物理形态的制品, 由于不再受到物理制品相关物理规律的限制, 其设计和构造过程体现出更大的开放性和灵活性.

另一方面, 在将存在于思维中的复杂抽象软件系统转变为实际可运行的软件系统的过程中, 人类个体智能在协同能力和信息处理能力上的局限性又为这一过程带来了极大的困难性. 复杂软件系统的开发是一种面向群体的智力密集型活动. 在将抽象软件系统落实的过程中, 需要不断凝聚共识, 确保开发者团队中的每一个体对于所要开发的软件系统的认知具有充分的一致性. 但人类个体智能在协同能力上的局限性, 为这种群体共识的形成带了天然的阻碍: 自然语言具有的模糊性和歧义性, 使得不同开发者之间很难准确理解他人思维中的信息; 即便采用具有充分表达能力的形式化语言进行交流, 形式化语言的学习和理解成本也无法满足实际软件开发的成本限制. 同时, 在这种落实过程中, 需要不断地为当前系统增加数量众多, 涉及多个方面, 且相互之间存在复杂关联的细节信息. 人类个体智能在信息处理能力上的局限性^[45, 46], 使得人类个体难以有效应对这种工作.

也就是说, 人类个体智能的特点和局限性是软件开发本质性困难难以克服的主要原因. 因此, 应对软件开发本质性困难的一种理论上可行的方式是求助于具有更高智能水平的智能体, 希望其智能水平能够有效克服软件的复杂性、一致性、易变性和不可见性等困难. 要使得这种方式具有实践可行性, 需要解决的一个关键问题是: 如何发现, 或者如何构造出这种具有更高智能水平的智能体? 互联网群体智能现象为这一关键问题指出了一种可能的解决方案, 即, 利用基于互联网的大规模人类群体协同, 形成一种具有更高智能水平的智能体.

2.2 软件开发组织方式的演化趋势

MIT Sloan 管理学院的 Thomas Malone 教授提出了一种关于人类群体组织方式的概念框架^[4]. 在这个框架中, 不同的群体组织方式按照其自身具有的“去中心化程度”(decentralization degree) 被组织在一个坐标轴中. 这里, 组织方式的去中心化程度指的是个体参与与其自身相关的决策活动的程度(参与程度越高, 则去中心化程度越高/中心化程度越低; 反之, 则去中心化程度越低/中心化程度越高). 图 1 给出了这种概念框架及其中存在的 5 种典型的具有不同去中心化程度的群体组织方式. 在该图所示的坐标轴上, 一种组织方式的位置越靠近左侧, 则表明其中心化程度越高; 越靠近右侧, 则去中心化程度越高. 表 1 对这 5 种群体组织方式的内涵进行了简要描述.

Malone 发现, 无论对于人类社会还是商业组织, 其组织方式的演化都体现出同一种模式: 从去中心化, 到中心化, 再回到去中心化. 图 2(a) 和 (b) 分别给出了人类社会和商业组织的组织方式的这种

表 1 具有不同去中心化程度的人类群体组织方式

Table 1 Organizational mechanisms of human collectives with different decentralization degrees

Organizational mechanism	Description
Centralized hierarchy	Decision-making authority originates from a few individuals, and a collective of individuals are organized as a hierarchy according to the degree of decision-making authority: the higher a level is, the fewer individuals are at this level; and the higher level individuals are at, the more important decisions they can make.
Loose hierarchy	Similar to the centralized hierarchy in that the decision-making authority originates from a few high-level individuals. The difference is that considerable decision-making authority is delegated to lower-level individuals.
Democracy	Decision-making authority originates from each individual, and decisions are made by voting, or by a few individuals who are elected by voting as delegates of other individuals in a collective.
Market	No individual is bound by a decision this individual does not agree. Each pair of individuals (as a buyer and a seller, respectively) can make their complementary decisions, subject only to their own wills and abilities, and also the rules of a market.
Individual independence	More decentralized than market in that individuals are independent of each other: no decision of an individual is bound to a trade between individuals.

演化模式. Malone 进一步指出, 形成这种演化模式的一个关键因素在于人类个体之间交互成本的持续降低. 以人类社会组织方式的演化为例. 在远古时期, 个体之间缺乏有效的交互方式, 只能作为独居动物独立生存 (自由度很高, 但生存难度也很大). 当个体之间掌握了面对面交互的有效方式后, 部落开始产生: 在部落中的个体, 相比较独居个体, 更容易在恶劣的自然环境中生存下来; 同时, 由于缺乏低成本的远距离交互方式, 部落只能占据一块较小的物理地域. 文字和印刷术的出现, 使得远距离的交互成为可能, 同时也促成了王国的出现: 与部落相比, 王国具有更大的领地, 也包含更多个体. 但是, 由于远距离交互的成本仍然很高, 王国通常采用集中式层级结构的组织方式, 以实现远距离交互的最小化与高效使用. 与部落相比, 王国中的个体通过牺牲个体的自由 (接受统治者的管理) 来确保稳定的生存环境. 电子通信技术的出现 (电报、电话、电视广播) 使得长距离交互的成本得到了有效降低, 也促成了现代国家的出现: 分布在不同地域的个体可以低成本地参与到决策活动中. 与王国相比, 现代国家中的个体在个体自由方面有了较大程度的提升.

需要指出的是, 人类个体之间交互成本不断降低的这种趋势仍在持续. 一个明显的证据即是互联网的出现和不断发展. 互联网提供了更低成本和更便捷的远距离交互方式, 正逐渐成为人类社会的一种基础性设施. 我们正在经历互联网对人类社会和商业组织造成的变革性影响; 这种影响在未来的很长一段时间内都会持续存在. 作为一种社会性或商业性活动, 传统的软件开发方式在这种变革过程中将很难“独善其身”.

考察软件开发的历史和现状, 可以发现, 软件开发组织方式的演化也遵循在人类社会和商业组织中发现的演化模式, 即, 从去中心化, 到中心化, 再回到去中心化 (见图 2(c)). 具体而言, 目前可以观察到软件开发经历的 3 个发展阶段: 原始软件开发, 工业化软件开发, 社会化软件开发.

- 原始软件开发. 在软件刚刚出现的早期阶段, 由于软件的规模很小、复杂度也不高, 软件由一个或若干个开发者即可开发完成. 在这一阶段, 软件仅仅应用在非常有限的领域中.
- 工业化软件开发. 随着软件作为信息化工具的定位逐渐清晰, 软件逐渐应用到各种问题领域中, 软件的规模和复杂性也在不断地提高. 在这个阶段中, 软件通常运行在相对独立的计算机上, 具有相

图 2 人类社会、一般商业组织及软件开发组织中组织方式的演化

Figure 2 The evolution of organizational mechanisms of (a) human societies, (b) businesses, (c) software engineering

对静态、封闭、可控的运行环境, 通常由软件公司内部的开发者团队进行开发. 这些开发团队通常采用集中式层级结构作为其组织方式. 正如 Brooks 所经历并指出的那样, 集中式层级结构在团队规模的可扩展性上存在缺陷: 当软件的规模和复杂性达到一定程度后, 简单地增加团队的规模 (向团队中增加更多的个体) 并不会带来软件开发效率和质量的相应提升 [2].

- 社会化软件开发. 随着互联网的出现和不断发展, 互联网逐渐成为一种全新的软件应用领域. 在互联网环境下, 软件的运行环境逐渐变得开放、动态和不可控; 软件也越来越多体现出分布、协同、泛在的特性; 软件的规模和复杂性提升到了一个新的高度. 这些变化也促使人们去寻找具有更好规模可扩展性的软件开发组织方式 [17, 47]. 例如, 通过使用去中心化程度更高的群体组织方式 (如, 松散式层级结构、民主、市场等) 来有效汇聚大规模互联网群体中每一个体的智慧和能力.

基于这种演化模式, 我们认为, 在互联网环境下, 为了应对软件规模和复杂性的不断增长, 在未来很长时间内, 人们将会持续探索采用更具规模可扩展性, 去中心化程度更高的组织方式来有效汇聚开发者群体中每一个体的智能. 其背后的逻辑是: 这种汇聚将会产生一种更高层次的智能 (即, 互联网群体智能), 其能够有效应对软件开发的本质性困难. 也就是说, 我们所要寻找的更高层次的智能, 不在别处, 而是蕴含在大规模的人类群体协同之中.

2.3 软件制品的结构特点

除了探索基于互联网的去中心化群体组织方式之外, 软件领域在近年来的另一个关注点是模型驱

动的软件开发^[48,49]. 其核心思想是将软件开发视为在一组模型之间的变换, 即, 以需求模型为起点, 逐渐向设计层和实现层模型变换. 不同种类的模型, 虽然在内容上具有不同的关注点, 但在结构上却体现出一些共性特点^[43]: 一个软件模型通常包括一组实体型和关系型元素, 而每一元素则由一组属性来进行更为具体的刻画. 例如, 面向软件问题域的概念模型 (一种需求层软件模型), 包含了问题域中存在的一组实体型概念, 这组实体性概念之间存在的关系型概念, 以及各个概念具有的属性; 软件体系结构模型 (一种设计层软件模型), 包含了一组构件 (component)、连接子 (connector) 以及对每个构件/连接子进行刻画的属性; 程序依赖图 (一种实现层的软件模型), 包含了一组语句节点、语句节点之间的控制依赖/数据依赖关系以及每个节点/依赖关系具有的属性. 模型驱动的软件开发提供了一种更加简洁和统一的视角来认识软件及软件开发活动.

我们认为, 模型驱动的软件开发和软件模型的结构特点为基于群体智能的软件开发提供了基本的可行性保证. 主要原因有 3 点. 第一, 在理论上, 基于模型驱动的视角, 软件开发可以被分解为一组关于特定模型的构造活动¹⁰⁾; 对于每一个模型, 其构造活动则可以进一步被分解为一组细粒度的建模操作 (如, 创建一个实体型或关系型元素, 为一个元素创建一个属性等). 这表明, 对于软件开发这一复杂问题, 其在求解方式上具有基本的可分解性 (即, 把一个复杂问题分解为一组较小粒度的子问题) 和可组合性 (即, 通过对一组较小问题解决方案的组合形成原始问题的解决方案). 基于这两种性质, 一个大规模的软件开发群体有可能通过并行、增量、持续迭代的方式进行模型构造活动, 形成复杂的软件模型制品, 并不断提高制品的质量. 第二, 自然界中存在的典型群体智能现象^[10,50], 与上述基于软件开发群体并行、增量、持续迭代的软件模型构造, 在原理上具有很强的类比性. 在宏观层次上, 对参与自然界群体智能现象的每一个体 (如, 一个蚂蚁、蜜蜂、细菌等) 而言, 在群体层次涌现形成的结构或模式具有超过其个体认知能力的复杂性. 但在微观层次上, 群体层结构或模式在其构成元素和元素之间的组合方式上并无任何神奇之处. 同时, 生物学家的观察和分析表明, 在大部分的自然界群体智能现象中, 并不存在一个领导者对群体中的每一个体的行为进行中心化的控制, 也不存在一个结构或模式的蓝图供群体参考. 相反, 群体层结构或模式的涌现过程实际体现为一个大规模的群体通过并行、增量、持续迭代的方式不断地将大量的基本元素放置/维持在合适的地点而逐渐形成的. 自然界的群体智能现象是一种存在于物理空间中针对物理结构或模式的群体智能现象; 相类比, 基于群体智能的软件模型构造则是一种存在于信息空间针对信息制品的群体智能现象. 前者的可行性和真实发生, 为后者的可行性提供了一定程度的正向支持. 第三, 在目前成功的开源软件和软件众包实践中, 软件制品的开发活动一般也体现为一种并行、增量、持续迭代的方式, 这在很大程度上印证了前两点分析的合理性.

3 基于互联网群体智能的软件开发: 3 种原始形态

为了真正实现基于群体智能的软件开发, 需要解决两个关键问题. 第一, 大规模的个体参与问题, 即, 如何吸引数量众多的人类个体持续参与到基于互联网群体智能的软件开发活动中. 大规模的个体参与是形成群体智能的先决条件. 第二, 大规模的信息激发与融合问题, 即, 如何有效地激发每一个体提供自己所拥有的潜在相关信息, 以及如何有效地对每一个体提供的信息片段进行融合, 从而在群体层次上形成高质量的软件制品. 大规模的信息激发与融合是形成群体智能的核心机理. 本节简要介

10) 在理想的模型驱动的软件开发中, 还存在另外一种重要活动: 基于规则的自动化模型变换/精化. 即, 基于预先设定的规则集, 通过自动化的方式把具有较高抽象级别的模型变换/精化为包含更多细节的模型. 但我们完全可以把预先设定的规则集也视为一种模型. 这样, 基于规则的自动化模型变换/精化活动就被间接归结为一种模型构造活动.

绍开源软件、软件众包以及应用程序商店这 3 种原始形态的基于互联网群体智能的软件开发实践, 并分析这 3 种原始形态对上述两个关键问题在实践中通过自发探索形成的解决方案。

3.1 开源软件

与软件开发所经历的 3 个发展阶段相对应, 开源软件^[18,20]的发展也经历了 3 个不同的时期。

- 田园期. 田园期对应于原始软件开发阶段. 在这一时期, 开源软件有实而无名: “开源软件”这个术语还没有出现; 但是, 当时的软件的确又是一种开源软件. 当时, 人们仅仅把软件作为硬件的一种附属物, 还没有人从商业的视角去看待软件. 对于软件开发而言, 相互分享各自软件的源代码是一件非常自然的事情. 人们完全被软件具有的神奇能力所吸引. 人类个体的想象力在软件开发中得到了极大的释放. 在这种想象力的释放中, 软件开发者也得到了极大的满足.

- 消沉期. 消沉期基本上对应于工业化软件开发阶段. 在这一时期, 在与闭源商业软件的竞争中, 开源软件¹¹⁾逐渐展现出其不成熟的一面: 开源软件过度强调软件的绝对自由主义, 排斥商业因素的介入. 这种竞争的结果是开源软件不再是一种受到广泛关注的主流软件开发实践. 但是, 开源软件并没有完全消失: 作为一种非主流现象, 开源软件开发仍然在不断地发展, 积蓄力量.

- 复兴期. 复兴期对应于社会化软件开发阶段, 也是我们目前正在经历的时期. 开源软件的复兴与互联网紧密相关: 互联网为现代开源软件开发中的“大规模的个体参与”、“大规模的信息激发与融合”等问题提供了技术上的可行性. 在这个时期, 开源软件成为了一种具有社会、政治、经济、技术等多重属性的独特现象. 成功的开源软件开发项目也展现出了群体智能的基本特点: 一个大规模的基于互联网的软件利益相关者群体, 通过协同构造出高质量的复杂软件系统, 展现出远超过单一开发者或传统软件开发组织的智能行为.

经过这 3 个时期, 开源软件逐渐形成了针对“大规模的个体参与”和“大规模的信息激发与融合”两个问题的实用型解决方案. 对于大规模的个体参与问题, 开源软件形成的解决方案具有 3 个关键点.

- 一种独特的所有权定义. 这里, 所有权定义指的是对“拥有某件物品”这一概念内涵的界定. 在开源软件中, 所有权的内涵不再是排他权 (the right to exclude), 而是发布权 (the right to distribute). 这种所有权被具体化为一组类似 GPL (general public license) 的开源软件许可, 其中显式声明了如下条款: 对开源软件的任何修改或改进都应该完整且不带任何限制地返回到开源软件开发社区中¹²⁾. 这种独特的所有权定义在法律层次上保证了任一个体对开源软件的贡献都不会被他人以排他性的方式占有. 这种所有权定义建立在软件是一种信息制品的基础上: 信息制品的特点使得软件的复制成本几乎为零; 两个人可以分别占有一段代码的两份拷贝, 而不会产生相互干扰.

- 动机无关的参与. 除了面向发布权的所有权中所规定的强制性条款外, 开源软件开发没有再为参与者附加任何其他限制. 具有任何动机的参与者, 无论是狂热的业余爱好者、关于软件自由的理想主义者、技术实用主义者, 还是具有商业目的的资本主义者, 只要遵守所有权规定的权利和责任, 都可以平等地参与到开源软件开发中, 而不会受到任何歧视性对待.

11) 需要指出的是, 在这一时期中也不存在“开源软件”这个术语, 存在的是“自由软件”这个术语. 一种常见的观点是将“自由软件”和“开源软件”作为两种不同的概念看待. 本文没有采用这种观点, 而仅仅将“自由软件”定位为“开源软件”发展过程中的一种过渡形态: 将“自由软件”中过于偏激的一些思想去除后, 其与“开源软件”并没有本质上的区别.

12) 目前也存在一些开源软件许可, 其甚至不再包含这种强制性条款. 我们认为, 这是开源软件独特的所有权定义得到广泛接受后一种更具包容性的表现. 但是, 这并不表明开源软件独特的所有权对开源软件不再重要: 如果所有的参与者只从开源软件中索取, 而不向其贡献, 那么, 开源软件将不再可能通过持续的演化不断发展.

- 参与和收益之间的正反馈回路. 对于任一个体而言, 其参与开源软件开发的程度越深, 就能够从有效的大规模信息融合中获得越多的收益; 这将进一步促使个体更深程度地参与到开源软件开发中. 这种正反馈回路也促使了如下情况的形成: 在一个大规模群体中, 个体通过自由贡献的方式协同提供了一种公共产品 (即, 一个开源软件系统); 与此同时, 每一个参与者都从这种公共产品中获得了超过其个体贡献的收益^[51].

对于大规模的信息激发与融合问题, 开源软件形成的解决方案包括 4 个关键点.

- 自愿型的任务分工与并行化的任务处理. 参与者自由选择其愿意且能够承担的软件开发任务. 因此, 一个任务通常会被最为合适的一个或一组参与者所承担, 避免了自上而下的强制性任务分工中可能产生的任务与其承担者之间的失配问题 (即, 任务被分配给了不合适的承担者). 同时, 通过并行化的任务处理 (多个承担者同时对一个任务进行处理), 通常可以从一组不同的候选方案中形成一个更优的方案 (通过竞争式的比较, 或通过协同式的取长补短).

- 持续改进. 由一个参与者贡献的方案, 可以激发其他参与者对其进行进一步的理解、评估与改进. 改进后方案又返回到开源社区中, 接受进一步地评估与改进. 因此, 即便一个初始的方案可能并不十分完善, 但在这种增量式的信息融合过程中, 其都会得到持续不断的改进和完善.

- 去中心化程度更高的决策机制. 开源软件开发通常面临两种类型的决策: (1) 采用何种类型的技术方案将一个粗粒度的开发任务分解为一组细粒度的子任务; (2) 哪一个参与者的贡献会被当前的代码库所承认和接收. 对于这些决策问题, 不同的大规模开源软件开发项目发展出了不同类型的去中心化决策机制. 例如, Linux 操作系统内核项目发展形成了一种基于松散式层级结构的决策机制: 一组决策者 (在该项目中被称为 maintainer) 通过自愿、自发、自然的方式形成了一个层级结构; 每一个决策者负责对一个或多个软件模块相关的问题进行决策, 然后将决策结果向上层决策者传递; 上层决策者进而决定是拒绝来自下层的决策结果, 或是同意并继续向上传递 (在存在上层决策者的情况下) 这些决策结果. Apache 项目发展形成了一种基于电子邮件投票的决策机制: 一个由项目长期贡献者形成的委员会, 采用最少法定人数 (minimum quorum) 规则, 对项目面临的各种问题进行决策 (包括委员会中新成员的加入问题); 其他贡献者也可以通过投票表达他们的意见, 但这些投票仅对委员会具有参考作用, 而不具有法律效力.

- 基于技术系统的大规模信息管理. 对于不同类型的信息, 开源软件开发项目通常采用不同的技术系统对其进行管理. 例如, 通过电子邮件系统或互联网论坛对开发过程中产生的讨论信息进行记录和管理; 通过缺陷追踪系统记录被发现的缺陷, 并对缺陷整个生命周期中的相关信息进行管理; 通过版本控制系统来记录、融合和管理由不同贡献者提交的源代码.

3.2 软件众包

众包 (crowdsourcing)^[16] 这个术语可以在两种意义上进行理解. 狭义上, 众包指的是 “一个公司或组织将原本由其雇员承担的功能, 通过公开征集的方式外包给一个未知 (通常具有较大规模) 的群体” (或者更简洁, 即 “将任务外包给群体”)^[52, 53]¹³⁾. 这也是众包这个术语在诞生时被赋予的含义. 广义上, 众包可以用来指代任何采用群体的力量求解问题的活动^[54]. 在这种含义下, 开源软件开发也可以被认为是一种众包现象. 这种广义理解使得 “众包” 成为了 “群体智能” 这一词语的近义词. 本文为了论述的明确性, 采用众包的狭义理解.

13) Crowdsourcing: a definition. http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html.

众包不是一种特定于互联网的现象. 朴素的众包现象可以追溯前互联网时代, 甚至是前计算机时代. 这些早期的众包现象可以大致被归为两类. 第 1 类众包关注于计算密集型问题的求解: 一个组织事先将一个原始问题分解为一组数量众多的简单计算任务, 然后通过公共媒体 (如新闻报纸) 征集求解这些任务的自由工作者, 并且按照每个参与者完成任务的数量向其支付报酬^[55]. 第 2 类众包关注于不可分解或很难被大规模分解的困难问题的求解: 一个组织将一个困难问题在公共媒体上发布以引起相关专家的注意, 并为提供最优或较优解决方案的一个或若干个参与者支付相应的奖金^[56].

互联网的出现极大地提高了众包活动的可行性和效率, 并使得众包发生了两个重要变化. 第一个变化是, 早期的第一类众包现象变化为关注于另一类特殊的可大规模分解问题的求解: 在这类问题中, 分解产生的是一组难于被计算机求解 (difficult for computer), 但易于被人类个体求解 (easy for human) 的任务. 这种变化的原因非常直观: 早期第 1 类众包所针对的计算密集型问题, 在自动化计算资源异常丰富的当前情况下, 已经没有必要再通过众包的方式进行求解. 另一个变化是, 在早期的众包现象中较为微弱的一种因素——参与众包的个体之间的协同——在互联网环境下的众包现象中变得越来越显著. 这种变化的技术原因也非常明显: 互联网为地理分布的一组个体之间的信息交换提供了一种高效、便捷、低成本的技术手段.

作为一种特定领域的众包现象, 软件众包也具有与一般众包现象相似的两种分类. 第 1 类软件众包的典型示例是众测 (crowdtesting)^[57], 即, 将软件测试任务众包给互联网上的测试者群体, 并根据每一参与者完成的测试任务的数量、质量、重要程度等因素向其支付相应的酬金. 目前, 互联网上存在数量众多的众测平台 (包括 Crowdsourced Testing, Passbrains, Testbirds, 99tests, Army Test 等¹⁴⁾). 第 2 类软件众包的典型示例是基于竞争的软件开发, 即, 将一个软件开发任务众包给互联网上的软件开发群体, 并为任务完成质量靠前的一个或若干个开发者支付相应的奖金. 这类众包软件开发的两个实际案例是“Matlab 在线程序竞赛”和“TopCoder”: 前者关注对一些趣味性难题 (如数独、填字游戏等) 的众包求解; 后者则关注对各种类型的软件开发任务 (包括需求分析、系统设计、编码等) 的众包求解.

对于大规模的个体参与问题, 软件众包提供了 3 种激励方式来吸引个体的参与. 第 1 种激励方式是金钱, 即, 支付给第 1 类众包中参与者的酬金, 或支付给第 2 类众包中获胜者的奖金. 第 2 种激励方式是赢得一项竞争性任务后获得的自我提升 (ego-boosting) 和声誉 (reputation): 前者是一种自我认可, 后者是一种来自他人的认可. 第 3 种激励方式是学习经历: 大多数参加竞争性众包任务的个体, 在很大程度上并不真正是为了赢得奖金, 而是参与其中所获得的学习经历 (通过学习他人提交的方案, 或通过与他人进行协同).

对于大规模的信息激发与融合问题, 第 1 类软件众包开发通过集中的方式将原始问题分解为一组对每一个体而言较易解决的子问题 (从而激发个体参与对这些子问题的求解), 并在所有子问题被解决后, 又通过集中的方式进行信息融合 (问题的分解方法即蕴含了解决方案的融合方法). 对于第 2 类软件众包 (即通过众包的方式求解一个不易分解的困难问题), 则可进一步分为两种情况. 在完全基于竞争的情况下, 信息的激发与融合问题并不存在. 在存在协同的情况下, 众包也发展形成了类似开源软件开发中基于持续改进的信息激发与融合. 例如, 一些软件众包项目通过“协同式竞争”的方式来激发个体的能力并对个体提交的信息片段进行融合, 即, 任何个体都可以对其他个体提交的解决方案进行改进, 并把改进后的方案在群体中公布, 以供其他个体对该方案进行进一步的改进. 另外, 对于一个解决方案, 在缺乏权威或定量的方法对其质量进行评估的情况下, 软件众包则通过投票或协同式评估

14) <https://crowdsourcedtesting.com>, <https://www.passbrains.com>, <https://www.testbirds.com>, <https://99tests.com>, <http://www.army-test.com>.

的方式来激发个体信息并对信息进行融合,即,每一个参与者都可以对候选解决方案进行投票或评价;然后,使用合适的统计方法对个体的投票或评价进行综合,从而形成群体对特定解决方案的评估结果。

3.3 应用程序商店

应用程序商店(如 Apple Store, Google Play 等)是一种基于互联网、针对特定软件运行平台的软件交易环境。在成功的应用程序商店中,可以观察到如下现象:在没有任何集中规划的情况下,涌现出大量持续演化的软件应用,从而不断满足一个大规模用户群体的需求。一个自然的问题是:在集中规划的情况下,是否也能形成这种现象?我们认为,在集中规划的情况下,几乎不可能形成类似的现象。主要原因在于,在一般意义上,对一个大规模生产系统中的各种要素进行有效的分配这一问题具有本质上的复杂性:这种复杂性超出了任何单一个体或传统组织的能力范围。针对应用程序商店,这种复杂性包括:如何从一个具有充分多样性且持续演化的大规模用户群体中捕获其需求,以及如何开发并演化一组数量众多的软件应用以满足用户群体的多样性和演化性需求。这种复杂性超出了传统软件工程方法的能力范围。一个进一步的问题是:这种现象是如何形成的?对于这一问题的答案,目前的基本认识是:市场机制在这种现象的形成中承担了重要角色。

对于大规模的个体参与问题,市场机制采用了一种共赢的策略来吸引软件应用的提供商和用户参与到应用程序商店中。对于软件应用提供商,他/她提供的软件应用数量越多且越能满足用户的需求,就能够从市场中获得越多的经济回报(通过直接或间接的方式)。对于用户而言,应用程序商店中出现了越多的高质量软件应用,他/她的个性化需求就越有可能被某个或某些软件应用所满足(通过付费购买或免费使用的方式)。

大规模的信息激发与融合问题,在应用程序商店这种场景下,具体化为如下问题:如何有效地激发并汇聚市场中每一参与者具有的信息,进而涌现形成数量众多的高质量软件应用?对于这一问题,市场机制提供的解决方案包含 3 个关键点。

- 信息的快速传播。有价值的信息可以通过有所有市场参与者形成的社交网络得到快速传播。其结果之一是:任何一个参与者都具有平等的信息获取权,进而基于获取到的信息进行独立的判断和决策。其结果之二是:那些具有更强信息汇聚能力和决策能力的产品提供商在市场中占据了领先地位。
- 劳动力的自由流动。一旦某个产品提供商认为其捕获了一组正确的用户需求且知道如何通过某种产品来满足这组需求,那么,下一步就是进行产品的生产。劳动力的自由流动,保证了产品提供商能够雇佣到一组合适的生产者来进行产品的生产。生产的过程,在某种意义上,就是将不同开发者具有的信息、知识或技能进行融合的过程。
- 持续的竞争。对每一个产品提供商而言,持续的竞争使得其不断地从市场中收集相关的信息,并作出相应的决策,从而不断提高其产品满足用户需求的能力。在全球层面上,持续的竞争实际上促成了一种分布式和协作式的大规模市场信息的融合方法。

需要指出的是,与开源软件和软件众包这两种软件开发实践相比,应用程序商店中的软件开发具有更高的抽象性。应用程序商店并不关注软件开发的具体技术细节(即在需求确定的前提下,采用什么样的技术手段开发出满足需求的应用程序),而更关注准确发现用户群体的多样性和演化性需求,以及劳动力(主要指具有不同技能的软件开发人员)在不同应用程序之间的自由流动(即获得更多市场关注的应用程序会吸引更多的劳动力向其流入;反之,市场关注度不断降低的应用程序则会导致劳动力的不断流失)。

同时,应该看到,开源软件、软件众包,以及应用程序商店三者之间并不存在本质上的冲突;相反,

表 2 基于互联网群体智能的软件开发成熟度分析

Table 2 The maturity analysis of software development based on Internet collective intelligence

Aspect	Theory/Technique	Platform	Practice
Maturity	★☆☆☆☆	★★☆☆☆	★★☆☆☆
Embodiment	(1) Exist theories about collective intelligence phenomena in nature. (2) Exist post-analysis/explanation about collective intelligence phenomena in software development, but lack theories that are commonly recognized. (3) Lack systematic understanding of key techniques.	(1) Lack systematic platforms to enable collective intelligence in software development. (2) Exist platforms based on general-purpose collaboration techniques (i.e., online forum, and email list). (3) Exist elementary platforms to support source code management and defect management.	(1) Exist many naturally-formed Internet-based software development practices. (2) Many software systems produced by these practices show their market success. (3) The success experience of is unreproducible. (4) Big difference with ideal collective intelligence based software development.

三者之间还存在一定程度的互补性. 例如, 应用程序商店中的软件应用可以采用开源或众包的方式进行开发; 复杂的开源软件项目可以采用应用程序商店的方式对其内部包含的子项目进行自由探索; 具有稳定商业收入的开源项目可以采用众包的方式实施特定的软件开发任务; 传统完全基于竞争的软件众包任务, 也可以尝试采用开源的方式进行协同式开发. 在目前的实践中, 三者的互补和融合在某种程度上已经发生或正在发生.

进一步而言, 传统闭源商业软件的开发与上述 3 种新型软件开发模式之间也不存在本质上的冲突. 把开源软件的开源范围、软件众包的众包对象、应用程序商店的劳动力市场限定为一个商业组织内部的开发者群体, 即可以将这 3 种新型软件开发模式应用于闭源商业软件的开发.

4 基于互联网群体智能的软件开发: 问题与挑战

基于互联网群体智能的软件开发, 为解决软件开发面临的本质性困难提供了一种创新性的途径. 但是, 虽然目前我们可以观察到很多成功的案例, 虽然有很多研究者对这些案例进行了具有深刻洞察力的事后分析和解释, 基于群体智能的软件开发系统的设计仍然仅仅是一种探索型的实践领域. 目前, 还没有一种成熟的原理、方法或技术能够确保软件开发中群体智能的可控形成和重复出现. 导致这种状况的主要原因在于 3 个方面的复杂性, 即, 群体智能现象的复杂性 (对于人类群体智能而言, 其不仅仅是一种技术现象, 还是一种具有心理、社会、经济等多重属性的复杂现象)、软件开发活动的复杂性 (软件开发涉及不同的阶段、不同的中间制品, 以及不同阶段/制品之间的复杂关系; 对其中涉及的每一种元素, 可能都需要寻找到一种合适的基于群体智能的实施、构造及维护方法), 以及将群体智能应用至软件开发的复杂性. 这种状况同时也蕴含了重大的研究机遇和挑战.

基于直接的观察, 我们认为, 目前开源软件开发、众包软件开发以及应用程序商店这 3 种基于互联网的群体化软件开发实践, 仅仅是基于互联网群体智能的软件开发的若干原始形态. 他们各自还存在一些明显的缺陷和问题. 无论是在理论技术, 还是在支撑平台或实践应用方面, 这几种原始形态都还没有达到足够成熟的阶段 (见表 2).

对于开源软件开发而言, 它还远远没有达到其所追求的完全去中心化的社会化软件开发. 大多数成功的开源软件项目依赖于一个小规模的精英群体对其顶层架构进行设计, 并对其版本发布进行严格的规划和控制; 大部分的参与者只是承担了边缘性的开发任务, 并没有真正参与到核心软件制品的开

发活动中.同时,开源软件目前更多关注对源码的管理和汇聚,对需求、设计等高层抽象制品的管理和汇聚还缺乏系统化的支持.

对于众包软件开发而言,其在形式上更加侧重金钱的激励机制在很多情况下并不能吸引到一个具有多样性的大规模开发者群体.而且,众包软件开发在实施过程中竞争多、协同少:只有完成质量最好或较好的开发者或开发者团队能够获得经济回报,其他参与者没有回报,人力资源浪费较为严重.同时,众包任务的完成时间成长尾分布:如果一个任务在短时间内无法完成,随着时间的推移,该任务受到的关注度会持续下降,从而使得该任务需要很长时间才能完成(甚至根本无法完成),时间效率太低.

对于应用程序商店而言,它是一种采用市场机制进行软件需求发现以及软件劳动力资源去中心化分配的实践,目前更多促进了大量高质量小规模软件的不断涌现.如何将这种机制用于大规模复杂软件系统的群体化开发,还有待尝试.

在更深的层次上,我们认为,要使得基于互联网群体智能的软件开发能够发展成为一种成熟的软件工程方法,还需要解决如下 4 个方面的基础性问题.

- 互联网人类群体形成的驱动因素.在人类历史的早期,人类群体的形成是一种自然选择的结果:以群体形式存在的人类个体比独居个体更容易生存.随着社会生产力的不断提高,生存对于人类个体而言,已经不在是一个关键的问题.其他因素在人类群体的形成中起到了更加重要的作用,这些因素包括:成本与回报、无私贡献、成就感、声誉、个性的释放、对美好事物的追求等.因此,设计一个人类群体智能系统的必要条件之一即是对这些新的驱动因素进行系统性的认识和理解,以便人类群体智能系统的设计者能够有效利用这些因素促成大规模互联网人类群体的形成.这些认知和理解与人类个体智能的心理学和社会学属性紧密相关.

- 群体智能的形成机理.目前,对于群体智能的形成机理具有很多不同的理解.一些研究者认为,群体智能是一种设计导致的机缘巧合(designed serendipity)^[58]:在一个大规模群体中,更容易发现求解一个复杂问题的最优个体组合;在这个最优组合中,每一个体具有的能力或信息具有最好的互补性;把这些具有互补性的能力或信息融合在一起,就能形成当前问题的一个最优解或较优解.有些研究者则认为,群体智能是一种对问题解空间的并行搜索机制,使得能够更有效地发现当前问题的最优解或较优解.还有一些研究者认为,群体智能与个体智能类似,也有其自己的心智能力(mental capability)^[59];这种心智能力是通过个体心智能力的恰当聚合所形成的.对群体智能形成机理的不同理解,反应了研究者对这一问题的不同观察/理解视角,对人类群体智能系统的设计提供了不同的关注点和决策依据.在这个方面,需要关注的核心问题是:这些不同的理解是对应于不同类型的群体智能,或者仅仅是揭示了群体智能的不同侧面;如果后者是正确答案,那么,是否存在一个关于群体智能形成机理的统一框架?进而,能否基于这种统一框架设计形成一个系统性的技术支撑平台,使得面向各类问题的人类群体智能系统的构造更加可控、便捷和高效.

- 针对不同软件开发问题的大规模分解机制.“分而治之”是求解问题的一般性策略.群体智能系统在很大程度上也能够从这种策略中获益:如果一个复杂问题能够被分解为一组大规模的简单问题,那么,将能够让更多个体以更低的成本参与到基于群体智能的问题求解活动中.但是,在传统软件工程方法中,分而治之的问题求解策略仅在一定程度上得到了应用.具体而言,传统软件工程中的问题分解,其对象是一个确定的小规模开发者群体,而不是一个具有更强不确定性且地理分布的大规模未知开发者群体.对于后者,实施问题分解的人员应该需要考虑更多的因素,包括:如何管理群体行为的不确定性,如何从群体获得的信息中排除冗余、错误及恶意欺骗,以及如何在问题分解中考虑这两种因素.更具实际意义的问题是,我们能否针对不同类型的软件开发问题,寻求到合适的大规模分解机制.

- 群体智能相关因素的度量.在已有的基于群体智能的软件开发实践中,通常可以观察到,软件开

发的质量和效率是在一种持续演化的过程中逐渐提升的。一个重要的问题是: 给定一个基于群体智能的软件开发项目, 在任何时间点上, 我们能否确定或评估当前软件开发活动的进展 (例如, 在当前情况下, 距离形成一个具有足够质量的软件产品, 还需要多长时间)。另一个问题则是: 如果采用经济利益作为激励机制, 那么, 对于不同的参与者, 我们应该支付给其多少酬金。为了回答这两个问题, 需要对群体智能的相关因素进行度量。具体而言, 这些度量需要考虑 3 个方面的因素: 群体行为的效率、群体输出制品的质量, 以及每一个体对群体输出的贡献程度。为了使得对这些因素的度量具有可行性, 一个基础性的研究问题是: 对基于群体智能的软件开发的效率和质量具有决定性、影响性或反映性的可观察指标有哪些, 以及这些指标之间具有什么样的关联关系?

5 基于互联网群体智能的软件开发: 理想形态

本节简要介绍我们目前对基于互联网群体智能的软件开发的理想形态的一些思考和探索。

建立基于互联网群体智能的软件开发理想形态的一个基本前提是需要对“面向人类的群体智能”和“互联网环境下人类群体智能的特殊性”这两个方面有一个较为明确的理解和定位。从工程应用的角度出发, 我们将人类群体智能定位为一种利用人类群体的力量求解复杂问题的途径。其中, “复杂问题”指的是依靠单一个体无法求解或有效求解的问题; “利用人类群体的力量”指的是把人类群体中每一个体拥有的能力或信息有效地激发并汇聚在一起。我们对于人类群体智能的定位, 主要建立在对人类文明这一现象的理解之上。我们认为, 人类文明的形成和不断演化是一种存在于物理空间¹⁵⁾的人类群体智能现象。进一步, 我们认为, 形成人类文明这一群体智能现象的基本原理是一种由信息的自由探索、融合、反馈 3 个基本活动形成的回路。在假定信息以书籍形式记录的前提下, 这一回路的 3 个活动大概表现为如下形式。在信息的自由探索活动中, 对于某一问题, 一组人类个体或小团体独立地进行各种尝试和探索。在信息的融合活动中, 一个考察过该问题实践现状的人类个体, 把他/她观察和想象到的各种方法以一种相对系统化的方式组织在一起, 形成一本书籍。在信息的反馈活动中, 这本书籍得到了某种程度的复制和传播 (复制的数量和传播的范围与当时的技术手段、问题的受关注程度以及书籍内容的质量紧密相关); 相关的个体或小团体在看到这本书后, 受到某种程度的启发, 对其当前的实践进行改进或变革。这 3 种活动持续不断的循环实施, 使得问题的解不断演化和优化。但是, 对于工程化问题的求解而言, 这种问题求解方式还不具有一般意义上的可行性。其表面原因在于: 这种方式不具有可控性, 而且通常体现出较大的时间跨度。其深层原因体现在 4 个方面: (1) 缺少足够多的参与者; (2) 信息在物理空间中传播速度太慢; (3) 信息的融合需要依赖于少数具有相关知识和创造性思维的人类个体; (4) 信息的反馈是一种被动式的反馈。这些原因, 使得物理空间中信息的自由探索 – 融合 – 反馈这一回路的循环速度太低, 因而使得我们很难在物理空间中观察到人类群体智能现象的太多实例。值得关注的是, 互联网的出现, 直接克服了上述 4 种深层原因中的前两种, 使得群体智能难以应用于工程化问题求解的状况正在发生改变。互联网使得大规模的人类群体在信息空间中更易形成, 也使得信息的传播速度显著提升, 进而使得信息的自由探索 – 融合 – 反馈这一回路的循环速度显著加快。这些方面体现了互联网环境下人类群体智能的特殊性, 也正是群体智能现象在互联网环境下不断涌现和蓬勃发展的主要原因。

进一步, 我们认为, 克服了“群体智能难以应用于工程化问题求解的 4 种深层原因中的前两种”的互联网群体智能, 还不是一种完全理想的形态。要达到互联网群体智能的理想形态, 还需要进一步

15) 与物理空间相对的是信息空间: 互联网群体智能是一种存在于信息空间的人类群体智能。

图 3 互联网群体智能的理想形态
Figure 3 Ideal Internet collective intelligence

克服这 4 种深层原因中的后两种, 从而打破信息融合对特定人类个体的依赖, 实现比被动式反馈更为高效的信息反馈, 最终进一步提高信息的自由探索 – 融合 – 反馈这一回路的循环效果和效率. 对于信息的融合, 当待融合信息的规模和复杂性达到一定的程度之后, 依靠人类单一个体的能力将不可能实现信息的高效和有效融合; 因此, 必须寻求更具可扩展性的自动化信息融合机制. 对于信息的反馈, 相比较被动式的反馈, 针对每一个体的能力或行为特点提供个性化和主动式的信息反馈, 将会更加高效. 基于上述分析, 我们认为, 理想形态的互联网群体智能是一种持续、增量、迭代的过程, 其中包含了一个由信息的自由探索、自动融合和主动反馈 3 个活动形成的回路 (见图 3). 在自由探索活动中, 每一个体自由提供其认为与当前问题的解决方案相关的信息片段; 在自动融合活动中, 软件算法对不同个体提供的信息片段进行自动化的融合; 在主动反馈活动中, 基于当前的融合结果, 软件算法对每一个体进行个性化和主动式的反馈, 以便激发每一个体进一步提供更多的有价值信息 (在融合结果正确的情况下, 可能会激发个体补充群体目前缺失的信息; 在融合结果存在错误的情况下, 则可能会激发个体指出其中的错误, 从而不断提高融合结果的正确性). 这一过程的持续、迭代、增量性体现在: 每当有任何一个体提供了新的信息片段 (不必同步地等待所有个体都提供了新的信息片段), 该信息片段就会被立刻融合到群体信息的当前融合结果中; 每当融合结果发生了变化, 就会立刻反映到对每一个体的反馈信息中. 这种理想形态的互联网群体智能具有 3 个重要特点: 基于演化的问题求解 (通过对解的不断演化逐渐提高解的质量)、基于大规模群体协同的演化 (通过大规模群体协同促成解的不断演化), 以及基于人机协同的大规模群体协同 (“人” 进行最具创造力的 “自由探索” 活动, “机” 进行计算密集型的 “自动融合” 与 “主动反馈” 活动; 通过人机协同促成大规模人类群体协同的形成, 实现群体层次上智能级别的显著提升).

基于对互联网群体智能的上述理解和定位, 我们初步设计了一种理想形态的基于互联网群体智能的软件开发基本框架 (见图 4). 下面, 为了论述方便, 我们用一种同步的论述方式对该框架蕴含的软件开发过程进行说明. 针对一个特定的软件开发问题, 首先将其发布给一个大规模的软件开发者群体; 然后, 群体中的每一个体自由提交对该问题的解决方案或解决方案片段; 接下来, 通过信息自动融合活动将片段信息融合为更具整体性的信息结构; 如果融合的结果形成了一个具有足够质量的解, 则输出该融合结果, 问题求解过程结束; 否则, 基于当前的融合结果, 对每一个体提供个性化和主动式的反馈; 每一个体在收到反馈后再次更新自己提交的信息片段; 然后进行新一轮的信息自动融合与主动反

图 4 基于互联网群体智能的软件开发基本框架

Figure 4 A framework for software engineering based on Internet collective intelligence

馈 (如果仍然没有形成可被接受的解).

对比这种理想形态的基于互联网群体智能的软件开发框架, 可以看到, 目前原始形态的基于互联网群体智能的软件开发实践, 存在的主要问题是缺乏自动化的信息融合和主动式的信息反馈机制. 以开源软件为例, 对于信息的融合问题, 其目前更多关注软件代码的融合; 对于需求和设计等非代码型软件开发信息的融合, 还完全处于人工阶段 (即由一个小规模精英群体对这些信息进行人工融合). 即便对于软件代码的融合, 虽然已经存在了自动化的代码版本管理工具 (仅具备初步的信息融合能力), 但是, 在决定将哪些开发者的贡献融合形成最终软件制品这一问题上, 大多数情况下仍然是由上述小规模精英群体进行决策和控制的. 对于信息的反馈问题, 目前的开源软件主要依赖于被动反馈, 即由参与者自己去开源项目的海量信息源中 (如, 论坛、源码库、缺陷管理系统等) 寻找自己感兴趣的相关信息或软件开发任务. 很少有开源项目能够主动地向参与者推送个性化的反馈信息. 可以看到, 由于缺乏自动化的信息融合和主动式的信息反馈机制, 开源软件中的“自由探索 – 融合 – 反馈”回路的循环效果和效率还存在很大的提升空间. 同时, 需要指出的是, 在目前的技术发展水平上, 大规模信息的自动化融合与主动式反馈已经具备了一定程度的可行性: 一方面, B2C 电子商务的持续发展促成了各种自动化信息推荐技术的出现和不断发展; 另一方面, 大数据技术的持续发展使得我们可以不再拘泥于基于规则的信息融合, 转而寻求更具有有效性和可扩展性的数据驱动的信息融合. 也就是说, 这种基于互联网群体智能的软件开发框架虽然具有很大程度的理想色彩, 但在核心技术上已经形成了一定的发展基础.

上述基于互联网群体智能的软件开发框架所蕴含的开发过程可以更为形式化地描述为如下形式. 存在一个复杂的软件开发问题 P , 其解或最优解为 S^* . 开发者群体的规模为 N . 在 0 时刻 (初始时刻) 的自主活动中, 每一个体提供的信息片段为 $Frag_i^0$, ($i = 1, 2, \dots, N$); 在 0 时刻的信息融合活动中, 这些信息片段被融合为一个解 $S^0 = \sum_{i=1}^N Frag_i^0$; 在 0 时刻的信息反馈之后, 进入 1 时刻的自主活动, 每一个体 i 根据接收到的反馈信息将其原来提交的信息片段更新为 $Frag_i^1 \leftarrow Frag_i^0 + S^0$; 然后, 在 1 时刻的信息融合活动中, 所有新的信息片段又被融合为一个新的解 $S^1 = \sum_{i=1}^N Frag_i^1$; 依次类推, 则可以

形成一个解的序列 $\{S^0, S^1, S^2, \dots\}$.

这一过程的理论可行性和有效性依赖于我们能否在如下两个基本理论问题上有所突破.

(1) 如何保证上述过程中形成的解序列的收敛性, 即 $\lim_{t \rightarrow \infty} S^t = S^*$.

(2) 在收敛性成立的情况下, 如何保证解序列的收敛效率, 即, 如何最小化时间 t , 其满足 $|S^t - S^*| < \Delta$. 其中, Δ 是一个预先设定的距离常量.

为了将这种基本框架落实到软件开发实践中, 需要进一步解决 3 个核心技术问题: 第一, 对于特定的软件开发问题, 如何对个体提交的信息进行结构化或形式化的表示, 使得软件算法能够对这些信息片段进行自动化融合; 第二, 如何对大量个体提交的信息片段进行有效和高效的融合, 形成更加有序、更有整体性的融合结果; 第三, 如何基于信息融合的结果对不同个体进行个性化的信息反馈, 以进一步激发其提供更有价值的信息. 对于这 3 个技术问题, 在国家自然科学基金重大项目“基于互联网群体智能的软件开发方法研究”的支持下, 北京大学、北京理工大学、南京大学、国防科学技术大学、北京航空航天大学等单位的学者, 正在进行一些初步的尝试和探索. 该重大项目旨在探索互联网环境下面向软件开发的群体智能的形成、度量与调控机理, 建立互联网环境下以大规模群体协同、智力汇聚、信誉追踪、持续演化为基本特征的新型软件开发模式. 限于篇幅原因, 本文不再对该项目的更多技术细节和研究进展进行介绍.

6 总结

我们正在经历一场软件开发方式的变革. 在这场变革中, 互联网承担了双重角色: 一方面, 互联网形成了一个开放、动态、持续演化的软件运行环境, 极大地增加了软件系统的规模和复杂性, 为软件开发带来了新的挑战; 另一方面, 在“将计算机互联”和“将物品互联”两种基础角色之上, 互联网正在展现出“将智能互联”的崭新角色, 促成了群体智能现象在各种问题领域中的出现和快速发展, 为解决软件开发面临的本质性困难提供了一种潜在的创新性解决方案. 本文从可行性、实践现状、问题与挑战等几个方面对基于互联网群体智能的软件开发进行了概要性的论述和分析, 并对其理想形态进行了初步的探索. 通过本文的论述和分析, 我们认为, 基于互联网群体智能的软件开发将会是一种重要的发展趋势.

需要指出的是, 基于互联网群体智能的软件开发, 在技术上目前还远未达到成熟可用的水平, 其进一步的发展和研究还存在很多不确定性和非共识性. 因此, 本文中一些有争议的内容更多体现为一家之言. 但我们希望通过这些论述和分析, 能够为软件工程领域的研究者对相关问题及研究实践现状的理解提供一种有价值的观察视角, 为软件工程在互联网环境下的创新性发展贡献一些可能有用的信息和观点.

参考文献

- 1 Brooks F P. No silver bullet: essence and accidents of software engineering. IEEE Comput, 1987, 20: 10-19
- 2 Brooks F P. The Mythical Man-Month: Essays on Software Engineering. 2nd ed. Boston: Addison-Wesley Professional, 1995
- 3 Brooks F P. The Mythical Man-Month: Essays on Software Engineering. 1st ed. Boston: Addison-Wesley Professional, 1975
- 4 Malone T W. The Future of Work: How the New Order of Business Will Shape Your Organization, Your Management Style and Your Life. Brighton: Harvard Business Review Press, 2004

- 5 Espinas A. *Des Sociétés Animales: études de Psychologie Comparée*. French ed. Ann Arbor: University of Michigan Library, 1877
- 6 Spencer H. *The Principles of Sociology*. London: Williams & Norgate, 1882
- 7 Thomas L. *Lives of a Cell: Notes of a Biology Watcher*. London: Penguin Books, 1978
- 8 Bonabeau E, Dorigo M, Theraulaz G. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford: Oxford University Press, 1999
- 9 Engels F. *Dialectics of Nature*. 1883
- 10 Camazine S, Deneubourg J L, Franks N R, et al. *Self-Organization in Biological Systems*. Princeton: Princeton University Press, 2003
- 11 Surowiecki J. *The Wisdom of Crowds*. Norwell: Anchor, 2005
- 12 Fisher L. *The Perfect Swarm: the Science of Complexity in Everyday Life*. New York: Basic Books, 2009
- 13 Herculano-Houzel S. The human brain in numbers: a linearly scaled-up primate brain. *Front Hum Neurosci*, 2009, 3: 31
- 14 Malone T W, Laubacher R, Dellarocas C. The collective intelligence genome. *MIT Sloan Manage Rev*, 2010, 51: 21–31
- 15 Lee J, Kladwang W, Lee M, et al. RNA design rules from a massive open laboratory. *Proc National Acad Sci*, 2014, 111: 2122–2127
- 16 Lakhani K, Garvin D, Lonstein E. TopCoder (A): developing software through crowdsourcing. Harvard Business School General Management Unit Case No. 610–032, 2010. <http://ssrn.com/abstract=2002884>
- 17 Herbsleb J D. Global software engineering: the future of socio-technical coordination. In: *Proceedings of Future of Software Engineering*, Minneapolis, 2007. 188–198
- 18 Kogut B, Metiu A. Open source software development and distributed innovation. *Oxford Rev Econ Policy*, 2001, 17: 248–264
- 19 Weber S. *The Success of Open Source*. Cambridge: Harvard University Press, 2004
- 20 Raymond E. The cathedral and the bazaar. *Knowl Technol Policy*, 1999, 12: 23–49
- 21 Ghosh R A, Glott R, Krieger B, et al. *Free/Libre and Open Source Software: Survey and Study*. Maastricht: University of Maastricht, 2002
- 22 Ipeirotis P G, Provost F, Wang J. Quality management on amazon mechanical turk. In: *Proceedings of the ACM SIGKDD Workshop on Human Computation*, Washington, 2010. 64–67
- 23 Ipeirotis P G. Analyzing the amazon mechanical turk marketplace. *XRDS Crossroads ACM Mag Students*, 2010, 17: 16–21
- 24 Tsai W T, Wu W, Huhns M N. Cloud-based software crowdsourcing. *IEEE Internet Comput*, 2014, 18: 78–83
- 25 Wu W, Tsai W T, Li W. An evaluation framework for software crowdsourcing. *Front Comput Sci*, 2013, 7: 694–709
- 26 Lévy P. *Collective Intelligence: Mankind's Emerging World in Cyberspace*. New York: Basic Books, 1999
- 27 Mayer-Kress G, Barczys C. The global brain as an emergent structure from the worldwide computing network, and its implications for modeling. *Inf Soc*, 1995, 11: 1–27
- 28 Heylighen F, Bollen J. The world-wide web as a super-brain: from metaphor to model. In: *Proceedings of Cybernetics and Systems' 96*, Vienna, 1996. 917–922
- 29 Bernstein A, Klein M, Malone T W. Programming the global brain. *Commun ACM*, 2012, 55: 41–43
- 30 Ahn L V. Human computation. Dissertation for Ph.D. Degree. Pittsburgh: Carnegie Mellon University, 2005
- 31 Quinn A, Bederson B. Human computation: a survey and taxonomy of a growing field. In: *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, 2011. 1403–1412
- 32 Lerner J, Tirole J. Some simple economics of open source. *J Ind Econ*, 2002, 50: 197–234
- 33 Mockus A, Fielding R, Herbsleb J. Two case studies of open source software development: apache and mozilla. *ACM Trans Softw Eng Meth*, 2002, 11: 309–346
- 34 Ye Y, Kishida K. Toward an understanding of the motivation of open source software developers. In: *Proceedings of the 25th International Conference on Software Engineering*, Portland, 2003. 419–429
- 35 Zhou M, Mockus A. What make long term contributors: willingness and opportunity in OSS community. In: *Proceedings of the 34th International Conference on Software Engineering*, Zurich, 2012. 518–528
- 36 Glance D. Release criteria for the Linux kernel. *First Monday*, 2004, 9: 52–64
- 37 Scacchi W. Free and open source development practices in the game community. *IEEE Softw*, 2004, 21: 59–66

- 38 Thomas C. Improving verification, validation, and test of the Linux kernel: the Linux stabilization project. In: Proceedings of the 3rd Workshop on Open Source Software Engineering, Washington, 2003. 133–136
- 39 Decker B, Ras E, Rech J, et al. Wiki-based stakeholder participation in requirements engineering. *IEEE Softw*, 2007, 24: 28–35
- 40 Nowak M, Pautasso C. Team situational awareness and architectural decision making with the software architecture warehouse. In: Proceedings of the 7th European Conference on Software Architecture, Montpellier, 2013. 146–161
- 41 Bruch M, Bodden E, Monperrus M, et al. IDE 2.0: collective intelligence in software development. In: Proceedings of FSE/SDP Workshop on the Future of Software Engineering Research, Santa Fe, 2010. 53–58
- 42 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]
- 43 Zhang W, Zhao H, Jiang Y, et al. Stigmergy-based construction of Internetware artifacts. *IEEE Softw*, 2015, 32: 58–66
- 44 Zhang W, Yi L, Zhao H Y, et al. Feature-oriented stigmergy-based collaborative requirements modeling: an exploratory approach for requirements elicitation and evolution based on web-enabled collective intelligence. *Sci China Inf Sci*, 2013, 56: 082108
- 45 Miller G A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychol Rev*, 1956, 63: 81–97
- 46 Cowan N. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behav Brain Sci*, 2001, 24: 87–114
- 47 Sinha V, Chandra S, Sengupta B. A research agenda for distributed software development. In: Proceedings of the 28th International Conference on Software Engineering, Shanghai, 2006. 731–740
- 48 France R, Rumpe B. Model-driven development of complex software: a research roadmap. In: Proceedings of Future of Software Engineering, Minneapolis, 2007. 37–54
- 49 Mellor S J, Clark A N, Futagami T. Model-driven development. *IEEE Softw*, 2003, 20: 14–18
- 50 Theraulaz G, Bonbeau E. A brief history of stigmergy. *Artif Life*, 1999, 5: 97–116
- 51 Hippel E V, Krogh G V. Open source software and the “private-collective” innovation model: issues for organization science. *Organ Sci*, 2003, 14: 209–113
- 52 Howe J. The rise of crowdsourcing. *Wired Mag*, 2006, 14: 1–4
- 53 Howe J. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. London: Crown Business, 2008
- 54 Doan A, Ramakrishnan R, Halevy A Y. Crowdsourcing systems on the world-wide web. *Commun ACM*, 2011, 54: 86–96
- 55 Grier D A. *When Computers Were Human*. Princeton: Princeton University Press, 2005
- 56 Dawson R, Byngall S. *Getting Results From Crowds: the Definitive Guide to Using Crowdsourcing to Grow your Business*. San Francisco: Advanced Human Technologies, 2011
- 57 Feng Y, Chen Z, Jones J, et al. Test report prioritization to assist crowdsourced testing. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, Bergamo, 2015. 225–236
- 58 Nielsen M. *Reinventing Discovery: the New Era of Networked Science*. Princeton: Princeton University Press, 2011
- 59 Heylighen F. Collective intelligence and its implementation on the web: algorithms to develop a collective mental map. *Comput Math Organ Theory*, 1999, 5: 253–280

Software development based on collective intelligence on the Internet: feasibility, state-of-the-practice, and challenges

Wei ZHANG^{1,2*} & Hong MEI^{1,2,3}

1. *Key Laboratory of High Confidence Software Technology (Peking University), Ministry of Education of China, Beijing 100871, China;*

2. *Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China;*

3. *School of Computer Science & Technology, Beijing Institute of Technology, Beijing 100081, China*

* Corresponding author. E-mail: zhangw.sei@pku.edu.cn

Abstract We are undergoing a revolution in the field of software engineering. In this revolution, the Internet plays a dual role: on one hand, the Internet provides an open, dynamic, and constantly changing environment for software, and greatly increases the scale and complexity of various software, bringing about challenges to the area of software engineering; on the other hand, based on the two roles of *internet-of-computers* and *internet-of-things*, the Internet is gradually exhibiting an entirely new role called *internet-of-intelligences*, and is facilitating the rapid growth of collective intelligence in diverse fields, providing opportunities to develop creative solutions to cope with the essential difficulties of software engineering. In this paper, we provide a brief introduction and analysis to software development based on collective intelligence on the Internet, from the three aspects of feasibility, state of the practice, and problems and challenges. We also demonstrate our preliminary thinking regarding the ideal form of software development based on collective intelligence on the Internet. We hope this study could provide a valuable viewpoint for software engineering researchers to understand the current and future status of software development in the context of the Internet, and provide some possible useful options to facilitate the creative development of software engineering.

Keywords software engineering, Internet, collective intelligence, open-source software, software crowdsourcing

Wei ZHANG was born in 1978. He received a Ph.D. degree in computer science from Peking University in 2006. He is currently an associate professor at School of Electronics Engineering and Computer Science, Peking University. His research interests include software reuse, requirements engineering, and collective intelligence based software engineering.

Hong MEI received a Ph.D. degree in computer science from Shanghai Jiao Tong University in 1992. He is currently a professor at Beijing Institute of Technology, and the director and a professor at the Key Laboratory of High Confidence Software Technology (Peking University), Ministry of Education of China. His research interests include software engineering and system software. He is an Academician of Chinese Academy of Sciences, and a Fellow of the IEEE and TWAS.