群智软件工程研究综述

徐立鑫[1] 吴化尧[1]

[1]（南京大学计算机科学与技术系 南京 210023）

（141270043@smail.nju.edu.cn）

A Survey of Collective Intelligence Software Engineering

Xu Lixin[1] Wu Huayao[1]

[1] （*Department of Computer Science and Technology*, *Nanjing University*, *Nanjing 210023*）

Abstract Collective intelligence software engineering aims to solve software engineering problems by techniques that exploit collective intelligence, which includes machine collective intelligence, human collective intelligence, and their combinations. As collective intelligence software engineering provides a new perspective for solving complex software engineering problems, it has become an important part of modern software engineering researches. This paper presents a survey of collective intelligence software engineering, which systematically reviews the applications of different collective intelligence inspired techniques on solving problems of requirements analysis, design, coding, testing and maintenance. Future researches and challenges in this research area are also discussed. The goal of this study is to establish a uniform framework of collective intelligence software engineering, and so to provide references for the interactions and transformations between collective intelligence techniques of different levels.

Key words collective intelligence; software engineering; crowdsourced software engineering; search based software engineering

摘要 群智软件工程旨在利用潜在高效的群体智能方法来解决软件工程问题，其中群体智能方法不仅包括机器群体智能，还包括人类群体智能以及人机结合群体智能。群智软件工程研究为解决复杂软件工程问题提供了新的思路，已成为现代软件工程的重要组成部分。本文以软件工程生命周期中的需求分析、设计、构造、测试和维护为主线，系统梳理和总结不同层次群体智能方法在上述软件开发活动上的应用。在此基础上，为不同层次群体智能方法间的相互借鉴与转化提供参考，并探讨群智软件工程的未来发展趋势和挑战。

关键词 群体智能；软件工程；众包软件工程；基于搜索的软件工程

中图法分类号 TP311

随着信息技术的飞速发展，现代软件的规模和复杂度正不断提高，开发高质量的软件已成为一项困难的工作；而软件需求的不断变化、开发周期的持续缩短、以及软件产品的快速迭代更是对软件的可重用性、可扩展性和可靠性提出了更高的要求和挑战。为了应对当前软件开发面临的困境，软件工程领域的学者近几十年来一直在不断探索新的软件开发方法，形成了面向对象软件工程[1]、面向服务软件工程[2]、网构软件工程[3]、可信软件工程[4]和大数据软件工程[5]等一批各具特色的软件工程方法。群智软件工程旨在利用群体智能方法来高效且富有创造力地解决软件开发各个阶段中遇到的各类复杂问题，是近年来软件工程领域研究的前沿和热点。

群体智能方法来源于智能行为在群体中的汇聚现象，其强调了在一群个体间的相互合作或竞争模式下，整个群体能表现出远超其中任何一个个体智能水平的智能能力[6]。这样一群个体既可以是没有智能、或仅具有相对简单智能的动物群体，也可以是人类群体本身。对应地，对群体智能的利用也就可分为下列三种不同的层次：机器群体智能、人类群体智能以及人机结合群体智能。

收稿日期：2019-06-30
基金项目：国家重点研发计划（2018YFB1003800）
通信作者：吴化尧（hywu@nju.edu.cn）

1) **机器群体智能。**机器群体智能方法利用模拟生物群体行为的进化优化算法[7]来进行复杂问题求解，其在软件工程中的代表性应用即为基于搜索的软件工程（Search Based Software Engineering，SBSE）[8]。在 SBSE 中，软件工程问题首先被形式化地转化为优化问题，随后机器群体智能方法就能在适应值函数的指导下自动搜索问题的最优解。
2) **人类群体智能。**人类群体智能利用大规模人类群体的协同来进行复杂问题求解，其在软件工程中的代表性应用即为众包软件工程（Crowdsourced Software Engineering，CSE）[9]。利用 CSE 解决软件工程问题需要三种角色的协同参与，包括问题的提出者、潜在解决问题的大规模分布式用户群体、以及一个促进各方交互的公共平台。
3) **人机结合群体智能。**人机结合群体智能是机器群体智能和人类群体智能的结合。对于某些软件工程问题来说，问题的某些方面可以使用计算机来自动优化，而在另一些方面上则需要手工进行求解，通过人机结合的方式有助于提高人们处理上述软件工程问题的有效性和自动化程度。

目前，在基于机器群体智能的软件工程（即 SBSE）和基于人类群体智能的软件工程（即 CSE）领域上已有很多研究工作，其中 Harman 等人[10]和 Mao 等人[11]已分别对上述两类群体智能方法在软件开发不同阶段的应用进行了全面细致的总结。然而，已有研究大多仅关注了单一层次的群体智能方法，忽视了不同层次间群体智能方法的内在联系，从而在一定程度上限制了群智软件工程方法的有效应用和发展潜力。

基于此，本文首先提出了应用不同层次的群体智能方法进行问题求解的统一框架；在此基础上，系统梳理和总结了各类群体智能方法在软件工程生命周期各项活动（包含需求分析、设计、构造、测试和维护）中的应用。我们希望本文工作能为不同层次间群体智能方法的相互借鉴和转化提供参考，从而进一步增强人们利用和优化群体智能方法来解决复杂软件工程问题的能力。

本文第一节介绍群体智能的基本概念和群智软件工程的统一框架；第二节从需求分析、设计、构造、测试和维护这五个阶段总结群智软件工程的研究现状；第三节讨论群智软件工程的挑战和未来研究方向，第四节对全文进行总结。

1 群智软件工程

1.1 群体智能

群体智能（Collective Intelligence）是在许多个体的合作和竞争中涌现的共享智慧，其在复杂问题求解上的优势主要在于群体对于个体智能的放大作用[12-15]。人们对于群体智能的认识很大程度受到生物物种的启示。Winston 等人[16]和 Schutter 等人[17]曾分别研究了蜂群和蚁群的协作行为，他们发现一群只有基本本能的个体在组成群体之后，整体上可以表现出远超其中任何一个个体的智能水平。这样一种生物群体现象随后启发人们提出了不同的进化优化算法（例如蚁群算法[18]、粒子群算法[19]和蜂群算法[20]），并将这些算法应用于解决难于通过传统方法构造和求解的现实复杂问题。

低智能生物体的群体行为让人们深刻认识到群体对智能的放大作用，大量学者因而开始考虑如何利用人群的智能解决某些难题。例如，Engelbart 在上世纪 60 年代就预言通过将人类的智力能力组织到更高层次的协同结构中可以放大人类的智力[21]；而 1714 年英国政府公开征集能够准确测量海上船只经度的方法则是众包的最早实践[22]。

然而，对人类群体智能的挖掘和利用出现的很晚，这主要是由于物理时空对大规模人类群体协同的限制，以使得人类个体之间的信息难于快速共享和传播[23]。互联网的快速发展极大地克服了这个问题，为人类群体智能的大规模应用奠定了基础，大量基于人类群体智能的应用也因此逐步出现。例如，免费协作、多语言的互联网百科全书 Wikipedia[1]；利用互联网大规模人类群体智能来辅助开发药物的 InnoCentive[2]；利用融合机器计算和人类计算[24]来高效解决任务的 Mechanical Turk[3]；利用游戏设计蛋白质模型，为抗逆转录病毒药物的设计提供新见解的游戏平台[25]；以及加强研究团体和社会中研究者的联系，丰富药物发现工作价值的众包平台[26]等都是人类群体智能应用的成功实践。

1.2 群体智能在软件工程中的应用

群体智能作为一类通用的问题求解方法，其同样被广泛应用于解决软件开发过程中面临的各类复杂问题，这一应用趋势经过逐步发展即形成了群智软件工程研究领域。目前，基于搜索的软件工程（SBSE）和众包软件工程（CSE）是该领域的两个重要研究方向，分别代表了对机器群体智能和人类群体智能的有

[1] en.wikipedia.org/wiki/Main_Page
[2] www.innocentive.com
[3] www.mturk.com

效探索和利用。

1.2.1 基于搜索软件工程

SBSE 最早可以追溯到 1976 年，Miller 最早使用遗传算法来解决浮点数测试用例的生成问题[27]。2001 年，Harman 和 Jones 系统性将软件工程问题转化为基于搜索的优化问题[8]，并利用遗传算法等群智优化算法进行问题求解，标志着 SBSE 开始成为软件工程中的一个新研究方向。

不同于传统软件工程在问题空间中通过算法构造出一个解，SBSE 将传统软件工程问题转化为优化问题，并以适应度函数作导向，利用群智优化算法在解空间中寻找问题的近似最优解。Harman 等人总结了应用 SBSE 的两个前提条件[10]：

1) 能将问题表述成形式化的优化问题

2) 能够定义出适应度函数（fitness function）

软件工程研究对象和解决方案都不是物理实体，这种虚拟性使得几乎软件开发生命周期各阶段的问题都可以形式化表示为优化问题。例如，在时间和成本的限制下，如何实现需求，实现哪些需求才能尽可能的满足要求？如何通过给定一个现有软件系统的模型、涉众需求的规范和有限的设计空间，找到一个能比现有解决方案更好满足需求的设计规范？以及如何减少自动测试中的冗余测试用例节省测试时间成本？通过数学抽象都可以形式化地将上述问题编码为优化问题。此外，软件工程中许多问题都有一组与之相关且丰富多样的软件度量标准，这些度量标准形成了良好的初始适应度函数候选集[28]。软件工程问题的这两个特点使 SBSE 的应用成为可能。

SBSE 的优势在于只要软件工程问题能够形式化表达就可以使用 SBSE 方法来解决问题，具有很强的普适性，并且在一定程度上低成本高效自动地完成软件工程任务，因此得到许多研究者地重视。调研中发现，经过长时间的发展，SBSE 领域已经有了将近 2000 篇相关文章。一些研究人员从不同的研究角度对 SBSE 领域进行了总结[10, 29, 30]。

1.2.2 众包软件工程

众包是一种新兴的协作模式，个人或组织从一个规模较大、相对开放且经常快速发展的群体中获得商品和服务。2006 年，Jeff Howe[9, 31]将众包定义为“公司或机构将需要员工完成的任务，以公开招募劳动力的形式外包给一个尚未界定(通常规模较大)的网络群体的行为”，以用来描述企业是如何利用互联网将工作外包给大众。

图 1 众包平台相关角色关系[11]

众包软件工程使用众包解决软件工程问题。通过互联网在线平台，软件工程问题被分解成一个个任务，分配给有能力的个体解决，利用人类群体智能的作用高效高质量给出解决方案。如图 1 所示，众包软件工程的工作流程涉及三个主要角色，基于互联网的众包平台，需求提出方和开发者。需求提出方可以是一个公司，公司利用众包平台完成公司业务可以加快软件开发周期，节省成本，个人也可以通过众包平台发布任务，帮助自己实现软件开发。开发者可以是个人也可以是一个团队，可以根据个人或者团队的能力选择相应的开发任务。基于互联网的众包平台是软件众包的核心，管理整个软件众包项目的业务逻辑，一个理想的软件众包平台应该有在线软件开发环境和工具、知识分享和协同工具、解决方案质量保证和改善工具以及项目管理工具[32]。

目前世界上最大规模的软件众包开发平台 TopCoder[4]于 2001 年建立，用户可以通过 TopCoder 发布软件设计、软件开发、软件测试等任务。此外，UDesignIt[5]、StakeNet[33]和 StakeSource[34]专注于需求获取与分析；DesignCrowd[6]可以利用群体智能进行软件设计；Bountify[7]、HelpMeout[35]等可以帮助用户进行代码编写和软件实现；而软件测试领域有最多的工具与平台，例如 uTest[8]、CrowdTesters[9]、99Tests[10]。

众包软件工程属于人类群体智能在软件工程中的实践形式，可以极大提高软件开发速度和交付软件产品的质量。与众包软件工程相关的研究也非常多，一些研究人员从不同角度对众包相关研究工作进行了总结[36-39]，其中 Mao 等人在 2016 年对众包技术在软件工程中的应用进行了系统的总结[11]。

1.2.3 群智软件工程统一框架

以基于搜索的软件工程（SBSE）为代表的研究旨在利用源自生物群体的进化优化算法来搜索软件工程问题的最佳解决方案，而以众包软件工程（CSE）为代表的研究则更加关注人类群体智能的高效汇聚

[4] www.topcoder.com
[5] www.marketingideasforprinters.com
[6] www.designcrowd.com
[7] bountify.co
[8] www.utest.org.cn
[9] www.easycounter.com/report/crowdtesters.com
[10] 99tests.com

和应用。这些方法虽然具有不同的群体智能表现形式，但从问题求解的统一的视角来看，对于特定的问题解空间，每个个体能感知和搜索的范围通常是有限的，而群体智能能通过个体间的交互作用来更有效地探索整个问题解空间，并最终找到问题的最优解或局部最优解。

图 2 给出了利用群体智能方法进行软件工程问题求解的统一框架。对于一个给定的软件工程问题，首先群体中的每个个体分别产生问题的解决方案，随后在群体智能的帮助下对已有解决方案进行评估，并在此基础上对解决方案进行优化以产生更好的解决方案。这一过程将不断循环迭代，直到满足某种终止条件，最后在整个流程中找到的最优解即作为群体智能的最终求解结果。

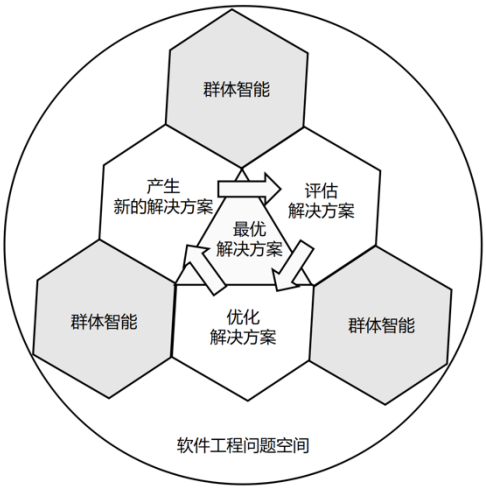

图 2 群智软件工程统一框架

产生解决方案、评估解决方案、以及优化解决方案是上述群智软件工程框架中的三个关键任务，这些任务既可以借助机器群体智能来实现，也可以借助人类群体智能来实现。当三个任务都使用机器群体智能来实现时，即代表了机器群体智能层次的群智软件工程（例如，使用蚁群算法自动化地构造、评估和优化测试数据[40]）；当三个任务都使用人类群体智能来实现时，即代表了人类群体智能层次的群智软件工程（例如，使用众包机制公开召集一群人类个体来构造、评估和优化软件设计[41]）；而当一部分任务由机器群体智能来实现，另一部分任务由人类群体智能来实现，即代表了人机结合群体智能层次的群智软件工程。

具体的，图 3 给出了利用机器群体智能进行软件工程问题求解的示意图。为了自动化地使用机器群体智能构造、评估和优化候选解，软件工程问题首先需要被结构化地编码为可以被计算和搜索的形式，同时需要给出候选解的形式化评估指标，即定义结构化的适应值函数；在此基础上，以进化优化算法为代表的机器群体智能方法就能在适应值函数的指导下自动化地演化候选解。

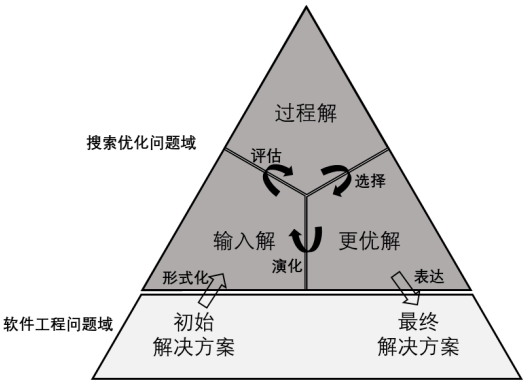

图 3 利用机器群体智能进行软件工程问题求解

图 4 给出了利用人类群体智能进行软件工程问题求解的示意图。其中，一个管理平台是群体智能有效涌现的基础，项目请求者在平台上发布软件工程项目和获得解决方案。软件工程项目在平台中将会经历产生解决方案、评估方案和选择一个比较好的解决方案的迭代过程，直到请求者获得一个满意的解决方案为止。在上述迭代过程中会产生许多的微任务，例如提交解决方案任务、评估其他人解决方案任务等。工人群体通过平台可以选择自己感兴趣的任务，提交任务结果。理想的平台还有工具支持，为大规模群体个体协作和通信奠定基础。

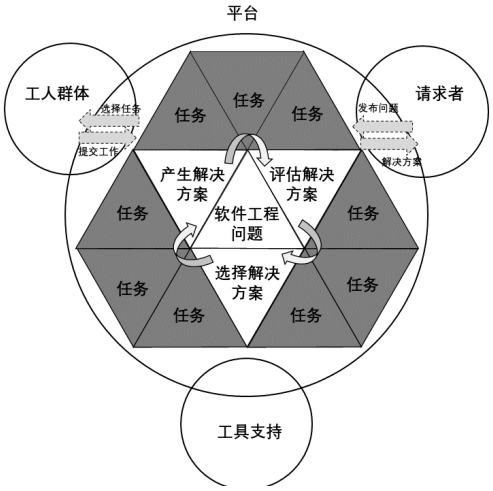

图 4 利用人类群体智能进行软件工程问题求解

对于某些软件工程问题，其解决方案可能难于形式化描述、或者难于形式化度量，这限制了机器群体智能的应用；而人类群体智能则要求所有任务都有人类个体来完成，这又缺乏对机器运算效率的有效利用。人机结合群体智能是对上述两者的结合，图 5 给出了

其对应的软件工程问题求解示意图。其中，机器群体智能既可以和人类群体智能结合，人机合力半自动化解决软件工程问题，也可以利用人群形成的知识库成为全自动化的间接人机结合群体智能来解决问题。

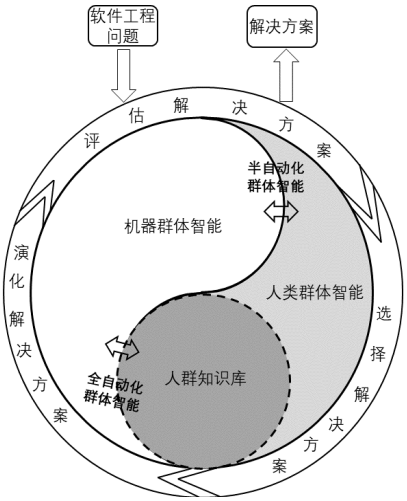

图 5 利用人机结合群体智能解决软件工程问题

基于机器群体智能、人类群体智能和人机结合群体智能的软件工程分别代表了利用群体智能进行软件工程问题求解的三种不同层次，它们统一于群智软件工程框架中。在接下来的章节中，我们将分别总结这些不同层次的群体智能在需求工程、设计、编码、测试和维护这五个阶段的研究和应用现状，力求为不同层次群体智能间的相互借鉴和转化提供参考。

2 需求工程

需求工程是通过分析客户需求，帮助技术人员理解问题域并定义目标系统功能特征的方法论。需求分析可以加强开发者对用户需求的理解，从而减少系统建设过程中可能出现的偏差与错误。需求管理能够很好追踪与处理需求变更问题，减少需求变更带来的耗费，提高生产效率。需求管理信息反映可以用于辅助项目决策。良好需求分析和管理是系统建设的基础，所以需求工程是软件开发生命周期中至关重要的一环[42]。

目前群体智能在需求工程中主要用于需求获取和分析，表 1 给出了不同层次的群体智能在解决具体需求工程问题上的应用。其中，机器群体智能在需求冲突协商[43]、需求交互[44-48]、需求选择[49-53]和需求规范[54]等需求分析活动中有较多应用；相比于机器群体智能，人类群体智能则主要在需求获取中发挥作用，如寻找需求涉众[33,34,55,56]、需求提取[57-61]和需求分析[34,55,56,62,63]等。目前，基于人机结合群体智能的需求工程研究较少，仅在需求规范[64]和需求优先级排序[65]中有相关研究。

表 1 群智需求工程相关研究

群体智能方法	需求工程问题	示例工作
人类群体智能	涉众确定	[33、34、56]
	需求提取	[57-61]
	需求优先级排序	[34、55、56、62、63]
机器群体智能	需求冲突协商	[43]
	需求选择	[49-53]
	需求交互	[44-48]
	需求规范	[54]
人机结合群体智能	需求规范	[64]
	需求优先级排序	[65]

2.1 涉众确定

需求获取指理解涉众的要求以及他们是如何使用新软件系统来支持他们的工作，这通常需要软件工程师和涉众的协同工作。如果项目需求分析的涉众群体不够多或者没有代表性，需求分析结果就会出现偏差，从而导致软件产品无法被用户接受。因此在需求获取之前应尽量全面地分析并寻找软件涉众。传统需求工程方法通常选取部分用户作为全体用户的代表，这会导致仅对有限的需求进行提取，从而难于适应互联网环境下大规模、多样和动态演化的用户群体和软件使用环境。

基于大规模人群涌现人类群体智能在寻找涉众任务中具有巨大优势。为了寻找和项目有关的所有涉众群体，Lim 等人[33, 34, 55]提出了 StakeNet 方法并开发了相关工具。StakeNet 根据涉众推荐涉众组成关系网形成的网络图，通过定义相关度量来计算涉众群体的优先级，在真实项目上的使用取得了良好的效果。Lim 等人[56]随后进一步提出了 StakeRare 方法，使用社交网络分析和协同过滤技术来识别大型软件工程中的需求并对需求优先级进行排序。

针对某些面向特殊用户的软件，找到足够多的特定类型用户是非常困难的。为了寻找到足够多的特定领域涉众，Wang 等人[57]根据具有特定领域知识的人群会在特定的时空间聚集的现象，提出了一种新型的参与者招募框架来以更有效的方式招募所需领域的参与者。

2.2 需求获取

需求提取是将各种形式地需求意见提取为可用于后续需求分析的规范需求文档的过程。人类群体智能方法充分利用了用户群体的主体作用，通过将需求

提取任务以一定方式众包出去，从而尽可能全面地感知和收集需求[60]。例如，Wen 等人提出了一种基于社会智能的网络化软件协同需求提取方法[59]，将需求语义概念定义为由群体参与生成的形式化需求描述，并通过社会参与对专家需求语义模型进行修正。同时该研究还为需求建模提供了帮助，具体是通过语义标记和验证完成了需求模型的实例化，此形式化需求语义模型可以直接导出到软件开发后续过程中。Hosseini 等人发现将众包用于需求提取可以适应软件需求提取的高要求[58]，该研究还深入研究了如何使用众包可以提高提取需求的质量。

在需求提取阶段，面临的困难主要在于从大量自然语言中提取需求时间成本很高，限制了需求提取的规模。Breaux 等人[61]采用任务分解的方式，将手工提取需求的任务众包给大规模群体，从而提高提取效率。

2.3 需求分析

需求分析包括提炼、分析和审查已经收集到的需求，以确保所有风险承担者都知道其中含义，并找到需求中冲突、错误和其它不足的地方。需求分析相关研究集中在需求优先级排序、需求分类、需求冲突协商、以及需求交互研究等活动。

目前，人类群体智能在需求分析中的应用主要是利用众包技术来进行需求优先级排序[34, 55, 56]。此外，Nascimento 等人[62]还利用众包对需求进行分类以分析用户对产品的真正期望；而 Liang 等人提出了基于 Web 2.0 的需求分析框架来辅助需求决策[63]，以降低用户参与需求分析的难度。

与人类群体智能相比，由于需求分析问题可以较容易地转化为结构化的优化问题，因而机器群体智能方法在该领域得到了广泛的应用。以需求分析中的下一版本问题为例。2001 年 Bagnall 等人[66]提出了下一版本问题来描述软件开发应该如何根据现有资源选择新的功能加入到软件系统中来满足客户需求，并应用了各种元启发式优化算法，包括贪婪算法、分支定界算法、模拟退火算法和爬山算法方法来进行求解。该方法是在资源有限的情况下最大限度地衡量利益相关者对公司的重要性。这种基于单目标配方的 NRP 是 SBSE 的第一次尝试。随后，由于功能选择时需要考虑多方面因素，例如客户之间的优先级关系、成本和需求之间的关系等，人们开始研究基于多目标优化的下一版本问题。其中，Brasil 等人[46]提出了一种多目标优化公式，在考虑客户满意度、需求之间相关性、业务价值和可用资源等多种因素的条件下确定系统理想的版本数量，实验中使用 NSGAII 和 MOCell[67]验证了公式的有效性。

NRP 问题还需要考虑项目资源和成本，在项目资源限制内找出一组需求满足不同涉众的诉求。Zhang 等人采用 Kiviat 图直观显示出预算压力对于多方涉众满意度的影响[49]，并且比较了 Two-Archive 算法、NSGAII 算法和随机算法在此问题下的性能，实验表明随着问题规模扩大，Two-Archive 算法的收敛性更好。项目预算不准确对 NRP 问题的最终结果有很大影响，Harman 等人在单目标或多目标需求优化问题基础上自动进行成本的敏感性分析[51]，决策者可以根据分析结果的可视化图表识别数据中的敏感范围。

在机器群体智能中群体优化算法的选择对解决问题有很大的影响。Souza 等人认为 NRP 问题可以看作是背包问题[45]，蚁群算法在这样的问题上表现非常好，因此作者对蚁群算法进行改进，使之可以解决存在互相依赖的需求的 NRP 问题。Kumari 等人结合量子计算和演化计算的特性，提出了一种量子启发的多目标演化算法来解决 NRP 问题中的需求选择问题[50]。Chaves 等人提出一个蜂群算法解决 MONRP 问题[52]，该方法可以生成高质量的需求集供工程师进行决策。José等人改进了早期的蚁群系统用于项目迭代中需求集的选择问题[53]，同样有不错的效果。

对于多客户相互竞争冲突的需求，在需求分析中需要协商解决冲突。Finkelstein 等人提出了三种公平性衡量公式[43]，并且利用遗传算法寻求需求公平性权衡的近似解。

需求分析中还需要关注需求交互问题[68]，即某些特定目标只有多个需求同时满足时才能达到。Zhang 修改了带精英策略的非支配排序的遗传算法（NSGA-II）[69]，使之可以用于解决考虑需求交互约束的多目标需求优化问题[44]，实验证明修改过的 NSGAII 在考虑 RI 时仍然可以保证解的质量和多样性。之后 Zhang 等人对算法进行了改进[47]，提高了算法的收敛性和最终解决方案的多样性。此外，Sureka 等人将 NPR 中的需求交互看作需求之间的协同作用[48]，分为积极协同作用和消极协同作用，研究了多目标演化算法对此类问题的性能，验证了多目标演化算法对 NRP 问题的适用性。

用户输入的需求可能是不准确不完全的，不能直接用作需求规范，Devries 等人结合符号分析和演化计算[54]，将需求进行分解，枚举所有必要的分解需求和确定分解需求的影响范围，非常具有挑战性。

人机结合群体智能能够使用人类智能来弥补机器智能的不足。Tonella 等人提出了一种基于交互的遗传算法来解决需求优先级排序问题[65]，使用人机结合群体智能来代替单纯机器群体智能，提高实际效果，该方法需要拥有相关知识的人员进行决策，实验证明

该方法有效性和鲁棒性均优于已有的交互式优先级划分技术 IAHP，并且可以容忍一定的人为错误率。Wever 等人采用协同演化算法通过确定性有限状态机半自动化寻找合适的形式化软件规范[64]，该方法需要用户对有限的训练数据进行标记。

此外 Sultanov 等人证明了群智优化也适用于需求跟踪问题[70]。Yue 等人为需求审查提出了适应度函数[71]，群智优化算法也可以用于自动需求审查分配问题。

3 软件设计

软件设计是从需求说明到软件具体实现到过渡阶段。人类群体智能在这方面的研究比较少，如表 2 所示，主要有软件概念建模[72]、UI 界面设计[41]和 web 网站页面设计[73]。

表 2 群智软件设计相关研究

群体智能方法	软件设计问题	示例工作
人类群体智能	软件概念建模 UI 设计 Web 网站设计	[72] [41] [73]
机器群体智能	基于构件软件设计 软件体系结构设计 软件产品线设计 服务软件设计	[74] [75-77] [78-83] [84、85]
人机结合群体智能	类设计	[86]

利用机器群体智能解决软件设计问题目前已有大量的应用，相关研究领域包括软件体系结构设计[75-77]、软件产品线设计[78-83]和面向服务的软件设计[84,85]等。上述这些软件设计方法在 20 世纪 90 年代以来已成为现代软件系统开发的基本准则，它们都充分利用了复用和模块化的设计思想，而其中如何通过选择并组合已有组件构造新软件系统的模式天然就契合机器群体智能解决问题的方式。例如 Lagerstrom 等人[74]根据非功能性需求、效用理论和涉众需求设计适应度函数，从组件库中选取新的组件对 UML 进行修改，搜索最佳解决方案。Räihä 从广泛的角度来考虑软件设计，不仅包括软件开发前期的软件设计，还包括软件维护和重构工程中的软件设计。总结了群智优化算法在软件设计中的应用[87]。Harman 等人系统总结了应用群智优化算法解决软件产品线工程中问题的研究[88]，为软件产品线的自动演化提供了方向。

具体地，在软件体系结构设计上，人们通常关注的是软件系统组织方式和整体结构，其目的是确定一个系统的主要构件以及构件之间的相互关系。机器群体智能在这一问题上已有很多应用，例如 Andrade 等人提出了 DuSE 方法[76]并开发了对应的 DuSE-MT[75]工具，该工具能够给出设计空间系统化表示元模型，并利用机器群体智能自动搜索帕累托最优的系统体系结构，为决策提供价值权衡；Li 等人则提出了 PETUT-MOO 工具[77]，可以在给定软件体系结构设计时自动优化和改进该设计的非功能性需求。

软件产品线架构（Product Line Architecture，PLA）指的是一个共性的体系结构以及共享构件的一组应用，其中每个应用都可以通过特征化来满足特定用户需求。在软件产品线设计上，Édipo Luis Féderle 等人提出一个工具 OPLA-Tool[78]，实现机器智能在多目标优化 PLA 任务上的辅助。现有优化搜索算子改变 PLA 的结构，可能违反分层体系结构，使得 PLA 灵活性和可维护性降低，为此 Mariani 等人引入考虑分层体系结构样式的搜索算子[79]，可以保证 PLA 风格不变并且与之前的方法相比有更好或等价的适应度函数。Wu 等人为了解决评价算法实际性能和确定实际使用中算法参数的问题，提出了一个寻找所有解决方案的三阶段算法[89]，可以促进决策者选择的决策。之后 Wu 等人还分享了使用该算法和群智优化算法解决实际问题的计算经验[80]，实验证明该算法能够帮助决策者确定算法的最优参数。PLA 是根据软件特征进行设计的，将特征模块化可以保证 PLA 的扩展，但是很少有优化算法考虑到特征的模块化，Colanzi 等人提出基于特征的算子用于 PLA 优化[81]，可以解决此类问题。Le 等人通过引入变异算子将设计模式引入 PLA 的设计优化过程中，提高了产生的解决方案的质量[82]。Colanzi 等人提出了新的基于搜索 PLA 直接表示方法[83]，比现有研究中的表示方法更有效率。

近年来，面向服务的软件体系结构也正受到人们的日益关注，其中每个服务实现一个特定的功能，并具有松耦合、可复用和技术异构等特征。在面向服务的软件设计上，人们通常面临服务组合和服务选择的问题，其中服务质量[90]（Quality of Service，QoS）是衡量解决方案的主要指标。机器群体智能同样在这一方面有所应用，例如云服务提供商提供的云服务及其相关定价模型各不相同，导致在成本最小化和保证 QoS 的条件下部署云服务的方案十分复杂，Ardagna 等人提出了一种 MILP 方法[84]，可以生成一个可行的初始解决方案用于自动搜索优化过程，该方案可以降低云应用程序的成本，并提高最终系统的质量；为了实现服务组合成所需系统的所有阶段自动化，Fanjiang 等人提出了一种服务组合自动化的方法[85]，

利用遗传算法满足服务组合过程中的所有功能性和非功能性需求。

此外，不同群智优化算法在软件设计中不同活动中也有不同的效果。Simons 等人提出了一种新型的交互式蚁群算法[91]，可用于软件开发生命周期早期的软件设计中，在计算预算有限的情况下，蚁群优化在优化度量方面比演化算法表现更好。

目前，机器群体智能在软件设计阶段已发挥了很大的作用，但是关于如何使用人类群体智能进行软件设计和应用的研究仍相对较少。其中，软件概念模型的质量对软件开发的成功有巨大作用，单个建模者无法建立高质量概念模型，Jiang 等人设计了一个工具[72]，可以使在线建模者突破时间空间限制进行协作，完成概念模型建模。对于没有经验的移动应用程序开发者来说，设计程序 UI 是十分困难的，Huang 等人利用群体的智慧建立了在线的程序应用线框图和 UI 设计示例之间的映射[41]，为开发者提供参考。Lasecki 等人提出了一个实时众包系统 Apparition[92]，通过理解需求方对软件功能的描述和画出的草图，在线群体帮组给出软件的原型设计。项目人员甚至可以通过 Apparition 表达自己的想法。Latoza 等人研究了众包软件设计中解决方案之间的关系[93]，其他人的解决方案可以帮助自己的解决方案演化，获得更好的解决方案，这为软件设计竞赛提供了好的改善方向。Nebeling 等人提出了轻量级 Web 信息系统开发方法[73]，网站设计的演化基于人群提供的数据和功能，网站所需组件的开发也由众包实现。

人机结合群体智能在软件设计问题上同样有一定程度的应用。例如，在面向对象编程中，Simons 等人[86]发现使用演化计算和软件代理的基于交互式搜索的人机结合群体智能方法进行类设计非常有前景。

表 3 群智软件构造相关研究

群体智能方法	软件构造问题	示例工作
人类群体智能	编码和调试	[35、94-96]
	程序优化	[97、98]
	程序修复	[99、100]
机器群体智能	程序优化	[101-106]
	程序自动修复	[107-112]
人机结合群体智能	程序优化	[113]
	程序自动修复	[114、115]

4 软件构造

软件构造是将设计模型实现为目标软件系统的过程，其中面临的主要问题是如何高效编写高质量的代码。如表 3 所示，人类群体智能方法在辅助调试[35,94-96]、程序优化[97, 98]和程序自动修复[99, 100]中都有应用。机器群体智能方法主要用来程序优化[101-106]和程序自动修复[107-112]。人机结合群体智能在程序优化[113]和程序自动修复[114, 115]中具有很好效果。

4.1 编码和调试

自 2010 年以来，已经广泛研究了在集成开发环境中使用人群知识来支持编码活动[11]，已经提出了一些工具和方法来利用人类群体智能帮助开发人员进行编码和调试。例如，经验不足的新手开发者在编码和错误调试上通常会浪费太多的时间，针对这一问题，Hartmann 等人[35]设计了面向静态语言的 HelpMeOut 推荐系统，通过展示其他程序员是如何纠正类似错误的示例来帮助用户调试错误和异常消息；随后，Mujumdar 等人给出了另外一种辅助工具 Crowd Debug，能够为动态解释的 Web 开发语言提供调试帮助[95]。此外，Fast 等人提供的 Codex 工具[94]能够记录程序员是如何使用编程语言的，新手程序员可以通过 Codex 掌握经典函数调用方式，避免出现不必要的程序语句。

这种利用人类群体智能的辅助工具需要预先收集大量代码方面的知识，如何建立对应的知识库因而也是相关研究中的一个重要问题。一般来说，人们通常从编程相关的问答社区中获取信息来建立知识库，但其中出现的代码片段往往不可直接运行。基于此，Terragni 等人[116]实现了将问答社区中代码片段自动合成可以直接编译的程序；Chen 等人[96]从 Stack Overflow 数据中建立代码匹配数据库，可以帮助开发者发现有缺陷的代码，并且提出修复缺陷的解决方案。

4.2 程序优化

利用群体智能对程序进行优化的研究主要关注编译优化。在人类群体智能层次上，Fast 等人设计了一个 Python 语言扩展 Meta[98]。开发者通过 Meta 可以共享自己写的函数，Meta 记录程序运行的数据，可以为开发者寻找代码优化和错误修复方案。Auler 等人通过收集 Web 客户端的性能数据，为 JavaScript 建立了在线编译标记参数推荐系统[97]，可以对给定的平台达到五倍的性能优化。

在人机结合群体智能层次上，Cochran 等人[113]提出了 Program boosting 方法来解决专业人士都难以解决的编程问题。例如在书写正则表达式时，通过熟练的开发人群提供问题的初始方案，通过演化来生成改进的解决方案，在这期间，未经训练的人群评估程序输出来帮助产生改进的方案，实验证明这样使得正则表达式精度得到了提升。

机器群体智能同样也被成功应用于程序优化。遗传优化（Genetic Improvement）是其中的代表性方法，其能通过自动搜索来改进软件的现有版本。在软件执行效率[101-106]、能耗[117-119]和内存消耗[120]等非功能性指标的优化工作中有很好的效果。

4.3 程序自动修复

程序修复包括故障定位、补丁生成和补丁验证三个阶段。Adriano 等人谈论了将复杂系统故障定位分解成可以众包的微小任务的可能性[99]，利用人类群体智慧进行故障定位还是很有前景的。Fryer 等人对使用奖励机制鼓励有能力的人发现软件漏洞的现象进行了系统的研究[100]，“漏洞奖励”能够帮助故障定位。

机器群体智能在程序自动修复中有很多的研究。Nguyen 等人提出了一种利用遗传编程(GP)自动进行软件修复的方法[109]。通过将修改集中在与 bug 发生位置相关的区域，该方法有效地将搜索空间复杂性最小化，从而提高了 GP 算法的性能。实验证明该方法能够在合理的时间内修复各种程序缺陷。

针对已有的应用程序，Weimer 等人提出了一种用于故障定位和修复缺陷的全自动方法 GenProg[107]。一旦发现程序错误，就使用一种扩展形式的遗传编程来演化程序变体，直到既保留了所需的功能又修复程序中的缺陷为止。Le 等人通过实验证明了该方法的有效性[108]。

大量自动修复相关研究集中在高级程序语言中，然而在汇编语言中进行程序自动修复有意想不到的好处。汇编语言层面上自动修复不仅不需要访问源代码，具有很强的普适性，而且所有修复都可以在更细粒度的级别上进行，有利于提升自动修复的效果。Schulte 等人描述了一种利用进化计算在汇编代码级自动修复遗留软件的方法[110]。在 Java 字节码和 x86 汇编程序上的实验展示了如何获得在保留所需程序特征的同时纠正缺陷的程序变体。

进化计算在程序自动修复中取得了很好的效果，但是在规模很大的系统中或者非常复杂的程序错误中，一般的适应度函数过于简单，算法的效率和准确性都会受到影响。Fast 等人提出适应度距离相关度量来评估程序[111]，实验显示了显著的程序改进和更平滑的修复过程。

除了改变适应度函数来提高程序自动修复的效率，还可以通过优化测试用例来提高效率。大多数自动程序修复技术都使用测试用例来验证生成的补丁的有效性。验证过程可能非常耗时，特别是当对象程序附带大量测试用例或一些长时间运行的测试用例时。为了提高效率，需要对测试用例进行约简。Qi 等人将回归测试优先级分析引入自动化程序修复领域，并提出了一种新的优先级分析技术 FRTP[112]，能够减少修复过程中执行测试用例的数量，大大提高自动修复的效率。

程序自动修复领域中，人机结合群体智能中人类群体智能的作用往往体现在为机器群体智能提供知识库，增强机器群体智能的效果。例如，上面提到的 GenProg 方法在应用时显示出了非常好的效果。但是由于突变操作的随机性，GenProg 会生成无意义的补丁。Kim 等人利用人机结合群体智能，提出了基于模式的自动程序修复技术[115]，将现有的人工编写的补丁作为知识库，通过学习总结出常见的修复模式为自动生成补丁服务。该方法生成的补丁比 GenProg 生成的补丁更容易被接受，且实验中的生成的补丁成功修复错误的概率更高。Nguyen 等人在使用 GI 对程序进行修复时也参考了人工开发的补丁来提高程序自动修复的效果[114]，人机结合群体智能在程序自动修复上有更好的效果。

5 软件测试和验证

群智软件测试是群智软件工程研究的一个重要分支。软件测试就是搜索软件中存在潜在错误的过程，这非常契合机器群体智能的使用方式；同时，让终端用户参与到软件测试过程中也是人类群体智能的研究关注点，因而目前已提出了很多基于群体智能的软件测试方法。在 SBSE 中，软件测试发展了很多年，已经成为一个非常大的领域。对整个基于搜索软件测试（Search Based Software Testing，SBST）领域进行详细的总结非常困难，很多研究学者就 SBST 中一个子领域做了比较详细的调研。例如，Anand 等人对自动测试用例生成领域进行了细致的梳理[121]，Malhotra 则着重对启发式搜索算法在自动测试数据生成中的应用进行了总结[122]。Sharma 调查了 GA 方法在解决软件测试中遇到的各种问题[123]。人类群体智能中，章等人对众包软件测试当前发展现状进行了总结，并展望了众包软件测试的未来发展[124]。

在基于群体智能的软件测试中，研究分为软件测试和软件验证过程。软件测试阶段，如表 4 所示，人类群体智能研究包括测试用例生成[125-127]、Oracle 问题[128]、GUI 测试[129, 130]、性能测试[131]和可用性测试[132]等，机器群体智能方法应用于测试用例生成[27, 40, 133-137]、测试用例排序[138]、Oracle 问题[139]、GUI 测试[140-143]、性能测试[144]、功能测试[145, 146]和回归测试[147]等和软件模型检测[148-151]，人机结合群体智能在测试用例生成[152]和 Oracle 问题[152, 153]中效果显著。在软

件验证中，人类群体智能在错误定位等方面有一些应用[154, 155]。

表 4 群智软件测试相关研究

群体智能方法	软件测试与验证问题	示例工作
人类群体智能	测试用例生成	[125-127]
	软件验证	[154]
	错误定位	[155]
	Oracle 问题	[128]
	GUI 测试	[129、130]
	性能测试	[131]
	可用性测试	[132]
机器群体智能	测试用例生成	[27、40、133-137]
	测试用例排序	[138]
	模型检测	[148-151]
	Oracle 问题	[139]
	GUI 测试	[140-143]
	性能测试	[144]
	功能测试	[145、146]
	回归测试	[147]
人机结合群体智能	测试用例生成	[152]
	Oracle 问题	[152、153]

5.1 测试用例生成

机器群体智能在软件工程领域应用的最早尝试就是解决软件测试问题。Miller 等人[27]使用遗传算法来自动生成浮点数测试数据，首次将测试数据生成问题转变为基于搜索的优化问题。随后，相关研究的发展十分迅速，基于搜索的软件工程领域中超过一半的研究都关注在软件测试[27]，其中包括遗传算法、蚁群算法等机器群体智能方法已被广泛应用于测试用例的生成[40, 133-137]、优先级排序[138]等各种问题。

Mahajan 等人提出一个模型[133]，可以用来检验遗传算法被用于使用数据流测试技术自动创建测试套件的性能。Rathore 等人在 GA 算法基础上使用禁忌搜索算法进行测试数据自动生成，在分支覆盖准则上比单一 GA 算法表现更优异[134]。Srivastava 等人对应用搜索技术创建测试用例进行了实验分析[40]，实验中 ACO 能比 GA 生成更高比例的有用测试用例。在具体应用场景中，对测试用例会有特别的需求。在回归测试中，时间往往不足以完成所以的测试用例，需要对测试用例的有用优先级进行排序，从而给出首先考虑的测试用例，Singh 等人实现了回归测试排序技术的算法形式[138]，使用 ACO 优化了排序结果。在路径测试中，Li 等人使用 ACO 自动化测试用例生成以完全覆盖代码[135]。类似的，Srivastava 等人也使用 ACO 自动生成测试用例来达到更好的代码覆盖率[136]。在特殊情况下，对于某些容易出错的程序路径需要重点测试，Rao 等人提出了一种适用于测试比较容易出错的路径的方法[137]，该方法提高了测试性能。

上述基于机器群体智能的自动测试用例生成节能省了大量的人力资源，代码覆盖率也很高，但是要改进自动化测试数据生成器实现更高层次代码覆盖率显得十分困难[156]。

人类群体智能同样也被应用于测试数据生成。Chen 等人提出基于谜题的半自动测试环境[127]，它将对象变异和复杂约束解决问题分解为小难题，供人类解决，较当前一些自动测试用例生成技术提高了代码覆盖率。Pham 等人对 Github 开展了一项研究[125]，GitHub 社区中有能力的人可以很轻松地解决别人代码库中出现的问题，从这个现象中发现了从社会性代码社区中生成测试用例的可能性[126]，但是这个想法还有待完善。

研究者也尝试使用人机结合群体智能来输出测试数据。Mcminn 等人曾提出一种从开发者、源代码和文档中抽取知识的自动测试数据生成方法[152]，从而通过利用这些知识来生成更加贴近真实应用场景、和更加容易判断测试结果的测试数据。

5.2 Oracle 问题

除了测试用例生成和排序外，群体智能方法在其它测试问题上也得到了广泛的应用。在测试用例的预期输出问题中，预期输出的判定通常需要人工来完成。Pastore 等人[128]曾开展一项实验来让人群判断程序断言和代码文档描述的行为是否一致，他们发现人类群体智能对解决预期输出问题有一定的帮助，但如何有效组织人类群体仍旧面临巨大的挑战。

使用人机结合群体智能可以有效降低人工成本。在现实代码中，检查软件行为常常是一项费力的手工任务。尽管人工 Oracle 的介入成本很高，但很少有研究调查如何让这个角色工作更容易、更省时。人工 Oracle 成本的一个来源是机器生成的测试输入固有的不可读性。特别是，自动生成的字符串输入往往是难以读取的任意字符序列。这使得测试用例很难理解，并且检查起来非常耗时。Afshan 等人提出了一种将自然语言模型引入基于搜索的输入数据生成过程中，以提高生成字符串的可读性的方法[153]。案例研究结果表明参与者在评估使用语言模型生成的输入时，记录的时间显著加快，部分测试输入评估的准确性也有显著提高。

Just 等人发现 Oracle 问题很难自动化解决，一些

全自动标准解决方案都有各种限制，极少情况下才能使用。利用底层函数或算法的约束来识别测试系统中的故障是比较有前景的 Oracle 方法[139]。根据输入数据的关系和程序底层函数或算法约束评估输出数据之间的关系间接解决 Oracle 问题。

5.3 GUI 测试

图形用户界面（GUI）是如今软件的重要组成组部分，它的正确执行是保证整个软件正确性的必要条件。但是在机器群体智能的应用中，创建一个可用于自动化测试用例生成的模型是困难的[157]，很少有工具能够帮助自动化测试过程。大部分研究只针对 GUI 测试的一个小阶段进行优化。

对于 GUI 测试，业界最常用的技术仍然是捕获和重放工具，它们极大简化了输入序列的记录和执行，但是不支持测试人员寻找故障敏感的测试用例。Bauersfeld 等人针对 GUI 测试序列生成问题，使用蚁群优化算法来搜索对故障敏感的测试用例[140, 141]。

在进行 GUI 测试时，常常会因为只研究用户界面而遗漏许多底层程序行为。Gross 等人开发出 EXSYST 原型工具[142]，使用遗传算法来进化 GUI 测试套件，以实现在探索用户界面的同时保证最大可能的代码覆盖率。

上述研究表明群智优化算法可以为 GUI 生成复杂的测试序列，但是测试序列有很强的随机性。对于希望用户在 GUI 中输入特定输入值的应用程序，测试效率会非常低。Salvesen 等人利用动态符号执行生成特定输入值[143]，实验中使用 DSE 对基于搜索的测试生成工具 EXSYST 进行拓展，成功增加了两个案例程序的测试代码覆盖率。

GUI 测试难以自动化，而人工测试 GUI 非常费时、昂贵且难以集成到连续的测试过程中。Dolstra 等人展示了使用人类群体智能解决 GUI 测试的可能[129]。实验证明，通过实例化测试系统的虚拟机，让测试人员通过互联网访问测试系统可以让大量参与者参与到测试过程中，从而在软件系统的连续测试中实现 GUI 的半自动连续测试，并且成本很低。Vliegendhart 等人在 Amazon 的众包平台 Mechanical Turk 上对一个多媒体应用程序进行 A/B 测试的例子[130]同样证明了人类群体智能在 GUI 测试上的巨大优势。

5.4 性能测试

性能测试的目的时发现影响系统 QoS 的因素，如响应时间、执行效率和可靠性等。群体智能在这方面的研究比较少。

在机器群体智能应用中，Gu 等人针对组合服务软件系统提出一种基于遗传算法的测试用例自动生成方法[144]，该方法根据系统的工作流程，将系统使用模式建模为基于性能需求的 QoS 敏感因子，采用群智优化算法自动寻找具有最大或最小 QoS 的测试用例。实验证明该自动测试数据生成方法生成的测试数据优于随机测试。

然而对系统建模仿真测试或者在受控环境中进行系统性能测试很难准确评估在极端异构环境中软件运行的性能。利用人类群体智能，将用户的真实使用性能数据收集起来用于性能测试评估能够帮助开发者团队识别和确定性能问题的优先级。Musson 等人在 Lync[11] 平台上的实验证明了该方法的有效性[131]。

并不是所有软件测试问题都可以用机器群体智能解决。例如，可用性测试是检查软件程序的人为因素和易用性等问题，人类群体智能基于人群的特点在这个领域有一些研究[132]，但是机器群体智能很难在这个领域发挥作用。总的来说，群体智能被广泛应用于解决各种软件测试问题，除上面提到的研究以外，功能测试[145, 146]、回归测试[147]等不同测试方法中遇到的问题都可以借助群体智能解决。

5.5 软件验证

软件测试的目的是发现程序中的错误，而软件验证则更加关注软件是否满足其所声明的功能性和非功能性需求。目前，机器群体智在软件验证中的应用主要集中于模型检测。例如，Alba 等人将蚁群算法应用于模型检测[148]，在状态空间中寻找反例。Banerjee 等人利用蚁群算法和粒子群算法来优化模型检测[149]。Katz 等人[150, 151]使用基于模型检查的 GP 在程序规格说明中验证和合成代码，模型检查用于提供每个阶段突变的适应度函数值。

软件验证一般由经过专业训练的工程师进行，如何让未经训练的普通人也能够进行软件验证工作，从而使用人类群体智能解决软件验证问题也是相关领域研究的重点。例如，Dietl 等人将验证任务转化成为小游戏，普通人熟悉游戏玩法便可以通过游戏证明软件的正确性[154]；Li 等人建立了一个系统 CrowdMine[155]，通过模拟实验证明未经训练的人群确实在一些软件验证工作如错误定位等有帮助作用。

6 软件维护

软件维护阶段主要任务包括软件演化、软件文档更新和其他一些软件工程更新方面的任务，其中

[11] products.office.com/en-us/previous-versions/microsoft-lync-2013

Mariani 等人介绍了应用于确定软件构件的最佳重构序列的基于搜索方法的共同特点，对这些方法进行概述，并确定了以后基于搜索软件重构的趋势和研究机会[158]。如表 5 所示，人类群体智能主要应用于软件演化[159-161]，机器群体智能在软件演化[162]和软件重构[163-165]及软件模块化[166-168]中都有研究，人机结合群体智能出现在软件重构[169]中。

表 5 群智软件维护相关研究

群体智能方法	软件演化与重构问题	示例工作
人类群体智能	软件演化	[159-161]
	文档维护	[170]
	软件本地化	[171]
机器群体智能	软件演化	[162]
	重构软件版本	[163]
	优化重构序列	[164、165]
	软件模块化	[166、167、168]
	系统迁移	[172]
人机结合群体智能	重构软件版本	[169]

6.1 软件演化

随着系统使用时间不断变长，系统环境发生变化，为了适应环境的变化，软件需要不断演化更新。在软件开发中，软件的上下文环境是很难监视的，Ali 等人提出 Social sensing[159]，利用终端用户监视软件上下文环境的变化，帮助软件再设计提出新的环境相关的需求，从而进行软件演化。He 等人提出建议模型[160]，鼓励用户参与到软件运行适应的流程中，用户可以提出和修改他们的意见，应用程序借鉴这些建议可以使自身运行时更好地适应环境。一些研究针对性更强，网站服务商必须让网站适应不同设备浏览的特点，提供能够适应各种设备的接口不仅十分有挑战性，而且成本很高，Nebeling 等人采用众包的方式，让用户参与网页接口自适应过程中，从而让网页可以适应广泛的使用环境[161]。

相比于上述人类群体智能的应用，机器群体智能在软件演化上的研究较少。Schulte 等人利用程序的“中性网络”和演化算法可以对软件进行自动演化[162]，这项研究将对现实软件系统的开发人员和维护人员有巨大帮助。

机器群体智能在软件维护中的应用主要集中在软件重构[173]和软件模块化[174]。

6.2 软件重构和模块化

软件重构是提升程序性能以减缓其由于更改而性能退化的过程。重构意味着通过修改程序来改进程序结构性，让程序更容易理解。SBSE 在软件维护阶段的研究重点就就是软件重构问题，相关研究主要关注下述两个方面[10]：一方面是利用群体智能方法将软件优化，形成重构后的版本[163,169]，另一方面是优化软件的重构步骤，使重构获得最好的效果[164,165]。

Cooper 等人利用遗传算法产生编译优化代码的最优序列，通过调整编译优化序列再对程序进行优化来减少编译代码的执行时间[163]。

重构旨在提高设计的质量，同时保留其语义。为重构提供自动支持是一个具有挑战性的问题。这个问题可以看作是一个优化问题，目标是使用一组重构示例找到合适的重构建议。如何评判重构建议的质量是非常困难的，人机结合群体智能对此有很好的解决办法。Ghannem 提出了一种采用交互式遗传算法来解决这一问题的方法[169]。该算法利用人的智力对产生的重构建议进行评价。

软件系统反复修改，但是底层设计并没有得到适当的调整，导致代码坏味[175]的出现。重构可以消除这些坏味，但是手工确定和执行有用的重构是一项艰巨的挑战。Kebir 等人提出一种基于遗传算法的基于构件软件自动重构方法[164]。该方法包括利用遗传算法搜索重构的最佳序列，检测并消除与构件相关的代码坏味。更普遍的软件系统中，需要消除的代码坏味严重性和代码需要被重构的重要性很难衡量，导致重构序列缺乏鲁棒性。Mkaouer 等人引入了一个基于 NSGA-II 的多目标健壮模型[165]，该模型在软件质量改进、代码坏味严重性和代码的重要性之间权衡以获取重构效果最大化。

软件模块化是降低软件复杂性的重要方法。软件模块化有助于软件实现可控、可维护和可扩展。Mancoridis 最早将机器群体智能算法用于软件的自动模块化[166]。该方法利用了传统的爬山算法和遗传算法，取得了不错的效果。随后 Mancoridis 等人还开发了一个聚类工具 Bunch[167]，实现了软件模块聚类。

在 Mancoridis 等人研究影响下，其他一些研究者也将软件模块聚类看作可以 SBSE 下的研究问题。例如 Harman 等人研究了将模块粒度、内聚性和耦合性度量作为适应度函数的影响[168]。

6.3 其他维护工作

除上述软件维护问题以外，群体智能还被应用在其他维护活动中。

例如利用机器群体智能实现遗留系统迁移到面向对象技术体系中的过程的完全或部分自动化，在代码中识别对象是整个过程的关键任务，Sahraoui 等人利用遗传算法在过程代码中识别对象解决了这一问

题[172]。

而在人类群体智能应用中，人群还被用来解决软件文档维护问题[170]和软件本地化问题[171]。

Jiau 等人提出了一种众包软件文档重用的方法，能够有效解决软件相关文档不足的问题[170]。研究团队在 Stackoverflow 平台上的研究证明了文档重用的可行性，同时实验还表明文档重用能够有效提高文档的覆盖率和质量。

软件本地化是指将软件产品的用户界面和辅助文档从其原产国语言向另一种语言转化，使之适应某一外国语言和文化的过程。Exton 等人发现众包是解决本地化的可行方法，众包使得软件本地化的决策从大型企业转移到服务用户，从而使软件可以有更多的语言可用，因此提出并描述了一种新的众包模式，它可以为实现突破语言障碍的“信息与知识的平等获取”提供一个平台。[171]

7 其它软件工程活动

表 6 其它群智软件工程活动相关研究

群体智能方法	软件工程活动	示例工作
人类群体智能	软件安全 用户支持 创意竞赛	［176、177］ ［178、179、180］ ［181］
机器群体智能	软件项目管理	［182、183］

群体智能在软件工程中的应用十分广泛，除了在软件开发生命周期五个阶段中有丰富的研究，在软件工程的其它方面也有很多的应用。如上表 6 所示，人类群体智能能够为软件安全[17,177]、软件用户支持[178-180]和软件创意竞赛服务[181]。机器群体智能则在软件项目管理的应用上拥有巨大优势[182,183]。

为了利用人类群体智能保证软件安全，Arellano 等人开发了一个基于众包的 Web 扩展，保证 Web 应用程序的安全性和完整性[176]。该扩展基于将用户看作不仅是程序的使用者，更是程序的贡献者的想法设计。类似的，Sharifi 实现了一个名为 SmartNotes 的系统，利用人群众包检测潜在的 Web 浏览安全威胁[177]。

而在软件用户支持方面，Chilana 等人发现即使在以用户为中心的设计和可用性方面做出了最大的努力，也不是所有的用例和用户交互中的细微差别都能在设计时预料到，实际中用户在寻找软件相关帮助也十分困难。因而设计了 LemonAid 帮助方法，允许用户从其他用户那里找到问题的答案，让软件支持能够被重用[178,179]。Chilana 等人之后还对 LemonAid 进行实例研究[180]。实验表明，LenmonAid 确实是有用的、直观的，并且值得重用的。

开展创意竞赛有益于众包的推广和软件开放创新过程。Leimeister 等人发现许多基于信息技术的创意竞赛不能吸引用户积极参与[181]。本研究强调了主办方的荣誉、奖励和专业知识作为激励的重要性。

软件工程项目管理关注的是软件生命周期不同阶段执行的复杂活动的管理，目标是优化软件生产过程以及这个过程产生的产品。机器群体智能对于这类多目标优化问题有很强的处理能力。Chang 等人最先将群智优化算法应用在软件项目管理中[182]。软件项目管理、调度和规划的应用引发了研究者的极大兴趣，随后出现了一批相关研究，例如 Kapur 等人使用遗传算法在有限的时间限制下为客户提供最优的产品发布人员和最优的服务质量[183]。

8 群智软件工程挑战及未来方向

8.1 机器群体智能

以基于搜索软件工程为代表的机器群体智能在软件工程中的应用已经是一个相当成熟的研究领域了。但是各种新的软件工程形式，如面向服务软件工程[2]、基于云的软件工程[184]、基于大数据的软件工程[185]的出现给 SBSE 带来了很多的挑战，SBSE 需要研究越来越广泛的软件工程问题。同时软件系统规模和复杂性不断增加，已有的群体优化算法计算性能越来越差，需要研究更加高效的群体优化算法应对更大规模的软件工作计算。

在研究越来越宽泛的软件工程问题上，机器群体智能还有很大的发展前景。首先，在已有的研究领域中还有许多问题无法使用机器群体智能来解决。如在需求工程中如何寻找尽可能多且具有代表性的用户群体、软件测试中涉及人因工程的可用性测试和 Oracle 问题都无法通过机器群体智能自动化解决。其次不断出现的新的软件工程领域问题等待机器群体智能提供高效自动的解决方案。

Harman 等人认识到只要能够解决软件工程问题及解决方案的表达方式和适应度函数的定义问题，就可以使用 SBSE 方法来解决软件工程问题[10]。为了能够利用机器群体智能研究越来越多的软件工程问题，一方面需要研究软件工程问题中的形式化表达问题，另一方面，SBSE 中评估与度量是一个重要的研究领域，一直在不断发展。软件工程中有许多评估方法，如何将这些评估方法转化为机器群体智能中需要的适应度函数仍然是一个巨大的挑战。

机器群体智能算法的发展需要新方法和新技术的推动来适应不断软件工程的发展。机器学习能够在

一定程度上弥补机器群体智能的不足，增强机器群体智能算法的性能，主要表现在：

1) 能够代替人工，加深机器群体智能方法的自动化程度
2) 优化群体智能算法性能

机器群体智能在某些问题上需要人工的帮助，这导致机器群体智能不能实现全自动化。融合机器学习算法和机器群智优化算法可以摆脱人计算能力限制，实现更高程度的自动化，提高生产效率。目前有利用机器学习代替人力提高机器群体智能自动化程度的相关研究，例如，Safdar 等人提出了一种多目标搜索和机器学习相结合的 SBRM 方法，可以不依靠人类智慧挖掘复杂的产品线规则[186]。实验结果表明，在适应度值、超容量和机器学习质量测量方面，SBRM 明显优于随机搜索。与真实数据挖掘的规则相比，SBRM 在准确率(18%)、召回率(72%)和 F-measure(59%)方面都有显著提高。Araújo 提出了一种利用机器学习技术对用户进行建模，并用其替换交互式遗传算法的 NRP 解决方案[187]。结果表明，该方法可以成功地将用户首选项包含在最终解决方案中。但是这个领域还尚未成熟，许多必须需要人力解决的问题还没有机器学习解决方案，同时在某些领域使用机器学习解决问题的质量不能得到保证。

机器群体智能方法不能全自动化解决更普遍的问题在于适应度函数定义不清、主观或难以量化，需要人的智力对过程解进行评估。Amal 等人介绍了一种基于神经网络的适应度函数在软件重构中的应用[188]。软件工程师通过遗传算法(GA)对建议的重构解决方案进行人工评估，然后人工神经网络(ANN)使用这些训练示例来评估剩余迭代的重构解决方案。机器学习方法可以利用训练的智能模型来代替适应度函数的作用，解决了没有合适适应度函数的问题。这是机器群体智能能够研究更普遍软件工程问题的新思路。

在第二点中，对于特定问题，机器群体智能使用者一直在寻找性能良好的群体优化算法[189]，然而这是一个非常单调和困难的过程。利用机器学习，可以自动搜索问题的算法空间，找到最适合的群智优化算法，提高工作效率。同样的，为同一问题的不同类别开发定制搜索算法的需求，阻碍了基于搜索的软件工程研究的发展[190]。未来机器群体智能的应用方式应该是在广泛的问题实例中学习搜索策略，提供一种单一的泛型方法。类似的超启发式搜索是将创新设计智能从人转移到机器的一种很有前途的方法，Mariani 讨论了超启发式算法在软件重构领域的巨大潜力[158]。另一方面，通过机器学习的数据分析预测能力能够为机器群体智能的应用提供新思路，提高机器群体智能算法的效率。例如在软件测试中，根据已有测试结果预测哪些剩余的测试用例会失败，哪些测试应该有更高的优先级。通过预测能力加快软件测试的速度。

如今软件工程以及机器学习都和大数据技术紧密相连。大数据为软件工程带来了许多挑战，大数据软件工程从新的角度重新审视软件工程，为机器群体智能的应用提供了借鉴。在此基础上，全自动机器群体智能软件工程解决问题的范式应该转变为数据驱动的软件工程方法。通过大数据技术从已有软件系统、开源软件和社区中采集海量的数据，将数据进行组织形成信息，之后对相关的信息进行整合和提炼，在数据的基础上经过机器学习训练和拟合，以机器群体智能为基础形成自动化的软件工程问题解决模型。

8.2 人类群体智能

人类群体智能从人群的竞争与合作中涌现。大规模流动人群是非常难以管理的，以至于利用人类群体智能解决问题充满了挑战。接下来从人类群体的组织管理、人群劳动力的激励机制和使用人类群体智能解决问题的质量保证三个方面阐述人类群体智能面临的挑战。

组织人类群体，使之有序、稳定、高效地涌现人类群体智能是使用人类群体智能解决问题的基础。Li 等人给出了软件众包的参考架构[32]。软件众包平台作为在线劳动力市场，是人类群体智能的架构核心。众包劳动力可以自主选择项目发起者要求的各种软件开发任务，需要有效促进人群中个体的协同作用。许多与劳动管理和项目治理相关的核心服务必须整合到平台中，包括专业排名、团队协作形式、任务匹配、奖励以及众筹。此外，每个人都应该能够使用为人群软件项目中的特定任务定制的设计和编码软件工具。

管理人群的群体智能还需要考虑软件工程项目中任务分解的粒度。Latoza 等人讨论了一些任务分解的注意事项[191]。细粒度的任务分解能够支持更大的并行性，但是在团队个体之间增加了通信开销，团队的协作与通信方式需要更加紧密，增加了在线众包开发的架构搭建难度；粗粒度的任务分解将导致任务的复杂性和工作量变大，不利于群体智能的发挥。同时在任务分解中，如何协调任务的边界和问题的边界问题。例如在确定如何重用外部库时，外部库的知识对很多微任务都有价值，不同任务的开发人员都需要阅读文档、查找实例得出解决方案，造成工作的重复。将类似的问题整合为一个特定的任务十分有必要。

人类群体智能中人群一般是单一群体，对这个群体进行任务分配。实际上多人群的人类群体智能能够

增加并行度，发挥出更大的作用。例如需求捕获和原型开发可以同时进行，一个人群用于收集需求，另一个人群为这些需求开发原型。人群的人员安排能够更合理地使用人群涌现的智能，在 Cochran 等人提出的 Program boosting 方法[113]中，存在两个不同的人群，在制作正则表达式的复杂事例中，熟练的开发人员提供问题初始方案的解，通过演化来生成改进的解决方案， 未经训练的人群对程序输出进行判定。在这期间，未经训练的人群评估程序输出来帮助产生改进的方案，这就是两个人群交互配合的良好例子。

以上提出了许多关于组织管理人群群体智能的想法，但是人类群体智能本身十分抽象，在使用时如何对群体协作程度进行度量与调控，保证群体智能的工作效率需要更多的研究。

人类群体智能的激励机制是群体智能涌现的动力，Varshney 实验证明了参与者动机对于激发参与性和确保可靠的交付平台至关重要[192]。Groen 等人在基于人群的需求工程中将参与者的自身动力分为 7 种[193]。最好的方式就是针对不同类型的个体给予相应的激励机制，但是这种方式在实现中十分的不切实际。

最后就是使用人类群体智能解决问题的质量保证问题。参与者短期参与项目，参与者的个人背景和工作能力各不相同，使用这种人群产生的群体智能来解决问题，容易产生质量问题[194]。Latoza 等人通过实验发现在软件设计中，如果参与者之间参考彼此的工作能够提高最终成果的质量[93]。除了互相借鉴以外，还有许多因素对项目的质量保证有影响，Li 等人通过对 TopCoder 的实例研究，确定了众包软件开发中 23 个和质量有关的因素[195]。许多质量相关工作有待评估，软件质量保证将会一直是人类群体智能软件工程未来工作的关注重点。

8.3 人机结合群体智能

当前人机结合群体智能是最理想的群体智能利用方式，结合了机器的高度并行性和人类智力的优点。人机结合群体智能的关键是人和机的相互关系处理，即人机结合群体智能的组织形式问题，如何针对人类和机器的特点设计并分配合适的任务？

现有的人机结合群体智能的出发点在机器群体智能在某些问题上存在局限性、需要人类智力的帮助才能解决问题。例如，在 Cochran 等人提出的 Program boosting 方法[113]中，通过专业人群生成群智优化算法的初始解，让未经训练的人群评估解的质量是人机结合群体智能的良好尝试。这种人机结合群体智能没有利用好人类群体智能，仅将其作为机器群体智能的补充。在 8.1 中，我们谈到利用机器学习的方法来替代交互式遗传算法实现全自动化机器群体智能方法[187]，类似的将半自动化人机结合群体智能转化为全自动化人机结合群体智能应该是未来的发展方向。

在这个层面上，人机群体智能需要的人的知识完全可以通过建立人群知识库来实现全自动化软件工程方法。群体智能作为软件工程基本方法，人群知识库建立和使用应该有统一的规范，对绝大部分软件工程问题都应该有借鉴意义，因此建立普遍适用的人群知识库时知识的表示和共享要能满足复用知识库的要求。人群知识库的建立和复用将极大缩减软件工程问题所需时间，提高软件生产力。不同知识库的相互补充使得全自动群体智能的性能不断提升，提高解决问题的性能。

在实际软件工程问题中应用群体软件工程方法，需要针对问题和解决方案的要求选择最合适的群体智能方法，根据已有条件权衡人机智能的使用平衡。

8.4 群智软件工程的应用和推广

群智软件工程为解决复杂软件工程问题提供了新的思路，已成为现代软件工程的重要组成部分。今后的研究应开展更多的经验研究，与传统软件工程方法进行系统对比。尤其针对云平台系统、移动应用、智能软件系统等新兴软件系统的特点，结合新的技术，研究新的群智软件工程方法，并开展实证研究和示范性应用。开发全面的群智软件工程工具集，为群智软件工程的推广奠定基础。

9 总结

本文从软件生命周期中的需求分析、设计、构造、测试和维护五个方面探讨了三个层次群体智能在软件工程领域的应用，并在此基础上讨论了不同层次群体智能未来的发展方向和可能的挑战。群体智能软件工程有着非常好的应用前景，但是同样存在一些问题亟待解决，群体智能软件工程将是软件工程的研究热点。

参 考 文 献

[1] Jacobson I. Object-oriented software engineering: a use case driven approach[M]. Pearson Education India, 1993

[2] Stojanović, Zoran, Ajantha Dahanayake, eds. Service-oriented software system engineering: challenges and practices[M]. Igi Global, 2005

[3] Lu Jian, Tao Xianping, Ma Xiaoxing, et al. Research on agent-based network architecture software model[J]. China Science: E, 2005(12):1233-1253 (in Chinese)

(吕建, 陶先平, 马晓星, et al. 基于 Agent 的网构软件模型研究[J]. 中国科学：E 辑, 2005(12):1233-1253)

[4] Liu X, Lang B, Xie B, et al. Software trustworthiness classification specification[R]. Tech. Rep. TRUSTIE-STC, 2009

[5] Madhavji N H, Miranskyy A, Kontogiannis K. Big picture of big data software engineering: with example research challenges[C]. Proceedings of the First International Workshop on BIG Data Software Engineering. IEEE Press, 2015: 11-14

[6] He Xiaoxian, Zhu Yunlong, Wang Mei. Overview of knowledge emergence and complex adaptability in swarm intelligence [J]. Information and Control, 2005, 34(5):560-566 (in Chinese)

(何小贤, 朱云龙, 王玫. 群体智能中的知识涌现与复杂适应性问题综述研究[J]. 信息与控制, 2005, 34(5): 560-566)

[7] Koza J R, Koza J R. Genetic programming: on the programming of computers by means of natural selection[M]. MIT press, 1992

[8] Harman M, Jones B F. Search-based software engineering[J]. Information and software Technology, 2001, 43(14): 833-839

[9] Howe, J., 2006a. Crowdsourcing: a definition. http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html

[10] Harman M, Mansouri S A, Zhang Y. Search-based software engineering: Trends, techniques and applications[J]. ACM Computing Surveys (CSUR), 2012, 45(1): 11

[11] Mao K, Capra L, Harman M, et al. A survey of the use of crowdsourcing in software engineering[J]. Journal of Systems and Software, 2017, 126: 57-84

[12] Heylighen F . Collective Intelligence and its Implementation on the Web:algorithms to develop a collective mental map[J]. Computational & Mathematical Organization Theory, 1999, 5(3): 253-280

[13] Krause S , James R , Faria J J , et al. Swarm intelligence in humans: diversity can trump ability[J]. Animal Behaviour, 2011, 81(5): 941-948

[14] Kaplan C A . Collective intelligence: A new approach to stock price forecasting[C]. IEEE International Conference on Systems. IEEE, 2001

[15] Krause J , Ruxton G D , Krause S . Swarm intelligence in animals and humans[J]. Trends in Ecology & Evolution, 2010, 25(1): 0-34

[16] Winston, M. L . Bee Work: The Wisdom of the Hive.[J]. Science, 1996, 272(5264): 967-967

[17] Schutter G D , Theraulaz G , Deneubourg J L . Animal–robots collective intelligence[J]. Annals of Mathematics and Artificial Intelligence, 2001,31(1-4): 223-238

[18] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]. Proceedings of the first European conference on artificial life. 1992, 142: 134-142

[19] Kennedy J. Particle swarm optimization[J]. Encyclopedia of machine learning, 2010: 760-766

[20] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005

[21] Engelbart D C. Augmenting human intellect: a conceptual framework (1962)[J]. PACKER, Randall and JORDAN, Ken. Multimedia. From Wagner to Virtual Reality. New York: WW Norton & Company, 2001: 64-90

[22] Sobel D. Longitude: The true story of a lone genius who solved the greatest scientific problem of his time[M]. Macmillan, 2005

[23] Zhang Wei, Mei Hong. Software development based on Internet swarm intelligence: feasibility, current situation and challenges[J]. China Science: Information science, 2017(12): 5-26 (in Chinese)

(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[J]. 中国科学:信息科学, 2017(12): 5-26)

[24] Harinarayan V, Rajaraman A, Ranganathan A. Hybrid machine human computing arrangement: U.S. Patent 7,197,459[P]. 2007-3-27

[25] Khatib F , Dimaio F , Cooper S , et al. Crystal structure of a monomeric retroviral protease solved by protein folding game players[J]. NATURE STRUCTURAL & MOLECULAR BIOLOGY, 2011, 18(10): 1175-1177

[26] Du L , Robles A J , King J B , et al. Crowdsourcing natural products discovery to access uncharted dimensions of fungal metabolite diversity[J]. Angewandte Chemie International Edition, 2015, 53(3): 804-809

[27] Miller W, Spooner D L. Automatic generation of floating-point test data[J]. IEEE Transactions on Software Engineering, 1976 (3): 223-226

[28] Harman M , Clark J . Metrics Are Fitness Functions Too[C]. International Symposium on Software Metrics. IEEE, 2004

[29] Sayyad A S, Ammar H. Pareto-optimal search-based software engineering (POSBSE): A literature survey[C]. 2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE). IEEE, 2013: 21-27

[30] Ilhem Boussaïd, Siarry P , Ahmed-Nacer M . A survey on search-based model-driven engineering[J]. Automated Software Engineering, 2017(9): 1-62

[31] Howe J. The rise of crowdsourcing[J]. Wired magazine, 2006, 14(6): 1-4

[32] Li W, Wu W, Wang H, et al. Crowd intelligence in AI 2.0 era[J]. Frontiers of Information Technology & Electronic Engineering, 2017, 18(1): 15-43

[33] Lim S L, Quercia D, Finkelstein A. StakeNet: using social networks to analyse the stakeholders of large-scale software projects[C]. Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering-Volume 1. ACM, 2010: 295-304

[34] Lim S L, Quercia D, Finkelstein A. StakeSource: harnessing the power of crowdsourcing and social networks in stakeholder analysis[C]. 2010 ACM/IEEE 32nd International Conference on Software Engineering. IEEE, 2010, 2: 239-242

[35] Hartmann B, MacDougall D, Brandt J, et al. What would other programmers do: suggesting solutions to error messages[C]. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2010: 1019-1028

[36] Yuen M C, King I, Leung K S. A survey of crowdsourcing systems[C]. 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing. IEEE, 2011: 766-773

[37] Zhao Y , Zhu Q . Evaluation on crowdsourcing research: Current status and future direction[J]. Information Systems Frontiers, 2014, 16(3): 417-434

[38] Kittur A, Nickerson J V, Bernstein M, et al. The future of crowd work[C]//Proceedings of the 2013 conference on Computer supported cooperative work. ACM, 2013: 1301-1318

[39] Chittilappilly A I , Chen L , Amer-Yahia S . A Survey of General-Purpose Crowdsourcing Techniques[J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(9): 1-1

[40] Srivastava P R, Ramachandran V, Kumar M, et al. Generation of test data using meta heuristic approach[C]. TENCON 2008-2008 IEEE Region 10 Conference. IEEE, 2008: 1-6

[41] Huang Y C , Wang C I , Hsu J . [ACM Press the companion publication of the 2013 international conference - Santa Monica, California, USA (2013.03.19-2013.03.22)] Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion - IUI \"13 Companion - Leveraging the crowd for creating wireframe-based exploration of mobile design pattern gallery[C]. Companion Publication of the International Conference on Intelligent User Interfaces Companion. ACM, 2013: 17

[42] Pohl K. Requirements engineering: fundamentals, principles, and techniques[M]. Springer Publishing Company, Incorporated, 2010

[43] Finkelstein A, Harman M, Mansouri S A, et al. “Fairness analysis” in requirements assignments[C]. 2008 16th IEEE International Requirements Engineering Conference. IEEE, 2008: 115-124

[44] Zhang Y, Harman M. Search based optimization of requirements interaction management[C]. 2nd international symposium on search based software engineering. IEEE, 2010: 47-56

[45] de Souza J T, Maia C L B, do Nascimento Ferreira T, et al. An ant colony optimization approach to the software release planning with dependent requirements[C]. International symposium on search based software engineering. Springer, Berlin, Heidelberg, 2011: 142-157

[46] Brasil M M A , Silva T G N D , Freitas F G D , et al. A Multiobjective Optimization Approach to the Software Release Planning with Undefined Number of Releases and Interdependent Requirements[J]. Lecture Notes in Business Information Processing, 2011, 102: 300-314

[47] Zhang, Yuanyuan, Mark, et al. Empirical evaluation of search based requirements interaction management[J]. Information & Software Technology, 2013, 55(1): 126-152

[48] Sureka A. Requirements prioritization and next-release problem under non-additive value conditions[C]. 2014 23rd Australian Software Engineering Conference. IEEE, 2014: 120-123

[49] Zhang Y , Harman M , Finkelstein A , et al. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation[J]. Information and Software Technology, 2011, 53(7): 761-773

[50] Kumari A C, Srinivas K, Gupta M P. Software requirements selection using quantum-inspired multi-objective differential evolution algorithm[C]. 2012 CSI Sixth International Conference on Software Engineering (CONSEG). IEEE, 2012: 1-8

[51] Harman M, Krinke J, Ren J, et al. Search based data sensitivity analysis applied to requirement engineering[C]. Proceedings of the 11th Annual conference on Genetic and evolutionary computation. ACM, 2009: 1681-1688

[52] Chaves-González, José M, Pérez-Toledano, Miguel A, Navasa A . Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm[J]. Knowledge-Based Systems, 2015, 83: 105-115

[53] José del Sagrado, IsabelM. Águila, Orellana F . Multi-objective ant colony optimization for requirements selection[J]. Empirical Software Engineering, 2015, 20(3): 577-610

[54] DeVries B, Cheng B H C. Automatic detection of incomplete requirements using symbolic analysis and evolutionary computation[C]. International Symposium on Search Based Software Engineering. Springer, Cham, 2017: 49-64

[55] Lim S L, Damian D, Finkelstein A. StakeSource2. 0: using social networks of stakeholders to identify and prioritise requirements[C]. 2011 33rd International Conference on Software Engineering (ICSE). IEEE, 2011: 1022-1024

[56] Lim S L , Finkelstein A . StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation[J]. IEEE Transactions on Software Engineering, 2012, 38(99): 1-1

[57] Wang H, Wang Y, Wang J. A participant recruitment framework for crowdsourcing based software requirement acquisition[C]. 2014 IEEE 9th International Conference on Global Software Engineering. IEEE, 2014: 65-73

[58] Hosseini M, Shahri A, Phalp K, et al. Configuring crowdsourcing for requirements elicitation[C]. 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS). IEEE, 2015: 133-138

[59] Wen B, Luo Z, Liang P. Distributed and collaborative requirements elicitation based on social intelligence[C]. 2012 Ninth Web Information Systems and Applications Conference. IEEE, 2012: 127-130

[60] Snijders R , Dalpiaz F , Hosseini M , et al. [IEEE 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC) - London, United Kingdom (2014.12.8-2014.12.11)] 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing - Crowd-centric Requirements Engineering[J]. 2014: 614-615

[61] Breaux T D, Schaub F. Scaling requirements extraction to the crowd: Experiments with privacy policies[C]. 2014 IEEE 22nd International Requirements Engineering Conference (RE). IEEE, 2014: 163-172

[62] Nascimento P, Aguas R, Schneider D, et al. An approach to requirements categorization using Kano's model and crowds[C]. Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2012: 387-392

[63] Liang P, Avgeriou P, He K, et al. From collective knowledge to intelligence: pre-requirements analysis of large and complex systems[C]. Proceedings of the 1st Workshop on Web 2.0 for Software Engineering. ACM, 2010: 26-30

[64] Wever M, van Rooijen L, Hamann H. Active coevolutionary learning of requirements specifications from examples[C]. Proceedings of the genetic and evolutionary computation conference. ACM, 2017: 1327-1334

[65] Tonella, Paolo, Susi, et al. Interactive requirements prioritization using a genetic algorithm[J]. Information & Software Technology, 2013, 55(1):173-187

[66] Bagnall A J, Rayward-Smith V J, Whittley I M. The next release problem[J]. Information & Software Technology, 2001, 43(14): 883-890

[67] Nebro A J , Durillo J J , Luna F , et al. MOCell: A cellular genetic algorithm for multiobjective optimization[J]. International Journal of Intelligent Systems, 2009, 24(7)

[68] Robinson W N , Pawlowski S D , Volkov V . Requirements interaction management[J]. ACM Computing Surveys, 2003, 35(2):132-190

[69] Deb K , Pratap A , Agarwal S , et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197

[70] Sultanov H , Hayes J H , Kong W K . Application of swarm techniques to requirements tracing[J]. Requirements Engineering, 2011, 16(3):209-226

[71] Yue T, Ali S. Applying search algorithms for optimizing stakeholders familiarity and balancing workload in requirements assignment[C]. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014: 1295-1302

[72] Jiang Y, Wang S, Fu K, et al. A collaborative conceptual modeling tool based on stigmergy mechanism[C]. Proceedings of the 8th Asia-Pacific Symposium on Internetware. ACM, 2016: 11-18

[73] Nebeling M, Leone S, Norrie M C. Crowdsourced web engineering and design[C]. International Conference on Web Engineering. Springer, Berlin, Heidelberg, 2012: 31-45

[74] Lagerström R, Johnson P, Ekstedt M. Search-based design of large software systems-of-systems[C]. Proceedings of the Third International Workshop on Software Engineering for Systems-of-Systems. IEEE Press, 2015: 44-47

[75] Andrade S S, de Araújo Macêdo R J. Toward Systematic Conveying of Architecture Design Knowledge for Self-Adaptive Systems[C]. 2013 IEEE 7th International Conference on Self-Adaptation and Self-Organizing Systems Workshops. IEEE, 2013: 23-24

[76] Andrade S S, Macêdo R J A. A search-based approach for architectural design of feedback control concerns in self-adaptive systems[C]. 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems. IEEE, 2013: 61-70

[77] Li R, Chaudron M R V, Ladan R C. Towards automated software architectures design using model transformations and evolutionary algorithms[C]. Proceedings of the 12th annual conference companion on Genetic and evolutionary computation. ACM, 2010: 2097-2098

[78] Féderle É L, do Nascimento Ferreira T, Colanzi T E, et al. OPLA-Tool: a support tool for search-based product line architecture design[C]. Proceedings of the 19th International Conference on Software Product Line. ACM, 2015: 370-373

[79] Mariani T, Vergilio S R, Colanzi T E. Search based design of layered product line architectures[C]. 2015 IEEE 39th Annual Computer Software and Applications Conference. IEEE, 2015, 2: 270-275

[80] Wu Z , Tang J . Designing and Reporting on Computational Experiments of Multi-objective Component Selection Algorithm[J]. International Journal of Information Technology & Decision Making, 2015, 14(02):375-394

[81] Colanzi T E, Vergilio S R. A feature-driven crossover operator for product line architecture design optimization[C]. 2014 IEEE 38th Annual Computer Software and Applications Conference. IEEE, 2014: 43-52

[82] Le Goues C , Yoo S . [Lecture Notes in Computer Science] Search-Based Software Engineering Volume 8636] A Pattern-Driven Mutation Operator for Search-Based Product Line Architecture Design[J]. 2014, 10.1007/978-3-319-09940-8(Chapter 6): 77-91

[83] Colanzi T E, Vergilio S R. Representation of software product line architectures for search-based design[C]. Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering. IEEE Press, 2013: 28-33

[84] Ardagna D, Gibilisco G P, Ciavotta M, et al. A multi-model optimization framework for the model driven design of cloud applications[C]. International Symposium on Search Based Software Engineering. Springer, Cham, 2014: 61-76

[85] Fanjiang Y Y , Syu Y . Semantic-based automatic service composition withfunctional and non-functional requirements in design time: A genetic algorithm approach[J]. Information and Software Technology, 2014, 56(3): 352-373

[86] Simons C L, Parmee I C, Gwynllyw R. Interactive, evolutionary search in upstream object-oriented class design[J]. IEEE Transactions on Software Engineering, 2010, 36(6): 798-816

[87] Räihä O. A survey on search-based software design[J]. Computer Science Review, 2010, 4(4): 203-249

[88] Harman M, Jia Y, Krinke J, et al. Search based software engineering for software product line engineering: a survey and directions for future work[C]. Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014: 5-18

[89] Wu Z , Tang J , Kwong C K , et al. AN OPTIMIZATION MODEL FOR REUSE SCENARIO SELECTION CONSIDERING RELIABILITY AND COST IN SOFTWARE PRODUCT LINE DEVELOPMENT[J]. International Journal of Information Technology & Decision Making, 2011, 10(05): 811-841

[90] Canfora G, Di Penta M, Esposito R, et al. An approach for QoS-aware service composition based on genetic algorithms[C]. Proceedings of the 7th annual conference on Genetic and evolutionary computation. ACM, 2005: 1069-1075

[91] Simons C L , Smith J , White P . Interactive ant colony optimization (iACO) for early lifecycle software design[J]. Swarm Intelligence, 2014, 8(2): 139-157

[92] Lasecki W S, Kim J, Rafter N, et al. Apparition: Crowdsourced user interfaces that come to life as you sketch them[C]. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, 2015: 1925-1934

[93] LaToza T D, Chen M, Jiang L, et al. Borrowing from the crowd: A study of recombination in software design competitions[C]. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE, 2015, 1: 551-562

[94] Fast E, Steffee D, Wang L, et al. Emergent, crowd-scale programming practice in the IDE[C]. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2014: 2491-2500

[95] Mujumdar D, Kallenbach M, Liu B, et al. Crowdsourcing suggestions to programming problems for dynamic web development languages[C]. CHI'11 Extended Abstracts on Human Factors in Computing Systems. ACM, 2011: 1525-1530

[96] Chen F, Kim S. Crowd debugging[C]. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, 2015: 320-332

[97] Auler R, Borin E, de Halleux P, et al. Addressing JavaScript JIT engines performance quirks: A crowdsourced adaptive compiler[C]. International Conference on Compiler Construction. Springer, Berlin, Heidelberg, 2014: 218-237

[98] Fast E, Bernstein M S. Meta: Enabling programming languages to learn from the crowd[C]. Proceedings of the 29th Annual Symposium on User Interface Software and Technology. ACM, 2016: 259-270

[99] Adriano C M, van der Hoek A. Exploring Microtask Crowdsourcing as a Means of Fault Localization[J]. arXiv preprint arXiv:1612.03015, 2016

[100] Fryer H, Simperl E. Web science challenges in researching bug bounties[C]. Proceedings of the 2017 ACM on Web Science Conference. ACM, 2017: 273-277

[101] Cody-Kenny B, Fenton M, Ronayne A, et al. A search for improved performance in regular expressions[C]. Proceedings of the Genetic and Evolutionary Computation Conference. ACM, 2017: 1280-1287

[102] Langdon W B, Harman M. Optimizing existing software with genetic programming[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(1): 118-135

[103] Langdon W B, Lam B Y H, Petke J, et al. Improving CUDA DNA analysis sof-tware with genetic programming[C]. Proceedings of the 2015 Annual Confe-rence on Genetic and Evolutionary Computation. ACM, 2015: 1063-1070

[104] Petke J, Harman M, Langdon W B, et al. Using genetic improvement and co-de transplants to specialise a C++ program to a problem class[C]. European Conference on Genetic Programming. Springer, Berlin, Heidelberg, 2014: 137-149

[105] Orlov M, Sipper M. Flight of the FINCH through the Java wilderness[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(2): 166-182

[106] White D R, Arcuri A, Clark J A. Evolutionary improvement of programs[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(4): 515-538

[107] Weimer W, Nguyen T V, Le Goues C, et al. Automatically finding patches using genetic programming[C]. Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009: 364-374

[108] Le Goues C, Dewey-Vogt M, Forrest S, et al. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each[C]. 2012 34th International Conference on Software Engineering (ICSE). IEEE, 2012: 3-13

[109] Nguyen T V, Weimer W, Le Goues C, et al. Using execution paths to evolve software patches[C]. 2009 International Conference on Software Testing, Verification, and Validation Workshops. IEEE, 2009: 152-153

[110] Schulte E, Forrest S, Weimer W. Automated program repair through the evolution of assembly code[C]. Proceedings of the IEEE/ACM international conference on Automated software engineering. ACM, 2010: 313-316

[111] Fast E, Le Goues C, Forrest S, et al. Designing better fitness functions for automated program repair[C]. Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, 2010: 965-972

[112] Qi Y, Mao X, Lei Y. Efficient automated program repair through fault-recorded testing prioritization[C]. 2013 IEEE International Conference on Software Maintenance. IEEE, 2013: 180-189

[113] Cochran R A, D'Antoni L, Livshits B, et al. Program boosting: Program synthesis via crowd-sourcing[J]. ACM SIGPLAN Notices, 2015, 50(1): 677-688

[114] Nguyen H D T, Qi D, Roychoudhury A, et al. Semfix: Program repair via semantic analysis[C]. 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013: 772-781

[115] Kim D, Nam J, Song J, et al. Automatic patch generation learned from human-written patches[C]. Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013: 802-811

[116] Terragni V, Liu Y, Cheung S C. CSNIPPEX: automated synthesis of compilable code snippets from Q&A sites[C]. Proceedings of the 25th

International Symposium on Software Testing and Analysis. ACM, 2016: 118-129

[117] Bruce B R, Petke J, Harman M. Reducing energy consumption using genetic improvement[C]. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. ACM, 2015: 1327-1334

[118] Manotas I, Pollock L, Clause J. SEEDS: a software engineer's energy-optimization decision support framework[C]. Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 503-514

[119] Li D, Tran A H, Halfond W G J. Making web applications more energy efficient for OLED smartphones[C]. Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 527-538

[120] Wu F, Weimer W, Harman M, et al. Deep parameter optimisation[C]. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. ACM, 2015: 1375-1382

[121] Anand S, Burke E K, Chen T Y, et al. An orchestrated survey of methodologies for automated software test case generation[J]. Journal of Systems and Software, 2013, 86(8): 1978-2001

[122] Malhotra R, Khari M. Heuristic search-based approach for automated test data generation: a survey[J]. International Journal of Bio-Inspired Computation, 2013, 5(1): 1-18

[123] Sharma C, Sabharwal S, Sibal R. A survey on software testing techniques using genetic algorithm[J]. arXiv preprint arXiv:1411.1154, 2014

[124] Zhang Xiaofang, Feng Yang, Liu Di, etc. Research Progress of Crowdsourced Software Testing[J]. Journal of software, 2018 (1) : 4 (in Chinese)
(章晓芳, 冯洋, 刘崭, 等. 众包软件测试技术研究进展[J]. 软件学报, 2018 (1): 4)

[125] Pham R, Singer L, Liskin O, et al. Creating a shared understanding of testing culture on a social coding site[C]. Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013: 112-121

[126] Pham R, Singer L, Schneider K. Building test suites in social coding sites by leveraging drive-by commits[C]. 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013: 1209-1212

[127] Chen N, Kim S. Puzzle-based automatic testing: Bringing humans into the loop by solving puzzles[C]. 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. IEEE, 2012: 140-149

[128] Pastore F, Mariani L, Fraser G. Crowdoracles: Can the crowd solve the oracle problem?[C]. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. IEEE, 2013: 342-351

[129] Dolstra E, Vliegendhart R, Pouwelse J. Crowdsourcing gui tests[C]. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. IEEE, 2013: 332-341

[130] Vliegendhart R, Dolstra E, Pouwelse J. Crowdsourced user interface testing for multimedia applications[C]. Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia. ACM, 2012: 21-22

[131] Musson R, Richards J, Fisher D, et al. Leveraging the crowd: how 48,000 users helped improve Lync performance[J]. IEEE software, 2013, 30(4): 38-45

[132] Schneider C, Cheung T. The power of the crowd: Performing usability testing using an on-demand workforce[M]. Information Systems Development. Springer, New York, NY, 2013: 551-560

[133] Mahajan M , Kumar S , Porwal R . Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach[J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(5): 1

[134] Rathore A , Bohara A , Prashil R G , et al. [ACM Press the Fourth Annual ACM Bangalore Conference - Bangalore, India (2011.03.25-2011.03.26)] Proceedings of the Fourth Annual ACM Bangalore Conference on - COMPUTE \"11 - Application of genetic algorithm and tabu search in software testing[J]. 2011: 1-4

[135] Li K, Zhang Z, Liu W. Automatic test data generation based on ant colony optimization[C]. 2009 Fifth International Conference on Natural Computation. IEEE, 2009, 6: 216-220

[136] Srivastava P R, Baby K. Automated software testing using metahurestic technique based on an ant colony optimization[C]. 2010 International Symposium on Electronic System Design. IEEE, 2010: 235-240

[137] Rao K K, Raju G, Nagaraj S. Optimizing the software testing efficiency by using a genetic algorithm: a design methodology[J]. ACM SIGSOFT Software Engineering Notes, 2013, 38(3): 1-5

[138] Singh Y, Kaur A, Suri B. Test case prioritization using ant colony optimization[J]. ACM SIGSOFT Software Engineering Notes, 2010, 35(4): 1-7

[139] Just R, Schweiggert F. Automating unit and integration testing with partial oracles[J]. Software Quality Journal, 2011, 19(4): 753

[140] Bauersfeld S, Wappler S, Wegener J. A metaheuristic approach to test sequence generation for applications with a GUI[C]. International Symposium on Search Based Software Engineering. Springer, Berlin, Heidelberg, 2011: 173-187

[141] Bauersfeld S, Wappler S, Wegener J. An approach to automatic input sequence generation for gui testing using ant colony optimization[C]. Proceedings of the 13th annual conference companion on Genetic and evolutionary computation. ACM, 2011: 251-252

[142] Gross F, Fraser G, Zeller A. EXSYST: search-based GUI testing[C]. Proceedings of the 34th International Conference on Software Engineering. IEEE Press, 2012: 1423-1426

[143] Salvesen K, Galeotti J P, Gross F, et al. Using dynamic symbolic execution to generate inputs in search-based GUI testing[C]. Proceedings of the Eighth International Workshop on Search-Based Software Testing. IEEE Press, 2015: 32-35

[144] Gu Y, Ge Y. Search-based performance testing of applications with composite services[C]. 2009 International Conference on Web Information Systems and Mining. IEEE, 2009: 320-324

[145] Wegener J, Bühler O. Evaluation of different fitness functions for the evolutionary testing of an autonomous parking system[C]. Genetic and Evolutionary Computation Conference. Springer, Berlin, Heidelberg, 2004: 1400-1412

[146] Bühler O, Wegener J. Evolutionary functional testing[J]. Computers & Operations Research, 2008, 35(10): 3144-3160

[147] Wagner M. Maximising axiomatization coverage and minimizing regression testing time[C]. 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2014: 2885-2892

[148] Alba E, Chicano F. Ant colony optimization for model checking[C]. International Conference on Computer Aided Systems Theory. Springer, Berlin, Heidelberg, 2007: 523-530

[149] Mahanti P K, Banerjee S. Automated testing in software engineering: using ant colony and self-regulated swarms[C]. Proceedings of the 17th IASTED international conference on Modelling and simulation (MS’06). 2006: 443-448

[150] Katz G, Peled D. Genetic programming and model checking: Synthesizing new mutual exclusion algorithms[C]. International Symposium on Automated Technology for Verification and Analysis. Springer, Berlin, Heidelberg, 2008: 33-47

[151] Katz G, Peled D. Model checking-based genetic programming with an application to mutual exclusion[C]. International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, Berlin, Heidelberg, 2008: 141-156

[152] Mcminn P , Stevenson M , Harman M . [ACM Press the First International Workshop - Trento, Italy (2010.07.13-2010.07.13)] Proceedings of the First International Workshop on Software Test Output Validation - STOV \"10 - Reducing qualitative human oracle costs associated with automatically generated test data[J]. 2010:1-4

[153] Afshan S, McMinn P, Stevenson M. Evolving readable string test inputs using a natural language model to reduce human oracle cost[C]. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. IEEE, 2013: 352-361

[154] Dietl W, Dietzel S, Ernst M D, et al. Verification games: Making verification fun[C]. Proceedings of the 14th Workshop on Formal Techniques for Java-like Programs. ACM, 2012: 42-49

[155] Li W, Seshia S A, Jha S. CrowdMine: towards crowdsourced human-assisted verification[C]. Proceedings of the 49th Annual Design Automation Conference. ACM, 2012: 1254-1255

[156] Lakhotia K, McMinn P, Harman M. Automated test data generation for coverage: Haven't we solved this problem yet?[C]. 2009 Testing: Academic and Industrial Conference-Practice and Research Techniques. IEEE, 2009: 95-104

[157] Memon A, Banerjee I, Nagarajan A. GUI ripping: Reverse engineering of graphical user interfaces for testing[C]. 10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings. IEEE, 2003: 260-269

[158] Mariani T, Vergilio S R. A systematic review on search-based refactoring[J]. Information and Software Technology, 2017, 83: 14-34

[159] Ali R, Solis C, Salehie M, et al. Social sensing: when users become monitors[C]. Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. ACM, 2011: 476-479

[160] He H , Ma Z Y , Chen H , et al. [ACM Press the 1st International Workshop - Hong Kong, China (2014.11.17-2014.11.17)] Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies - CrowdSoft 2014 - How the crowd impacts commercial applications: a user-oriented approach[J]. 2014:1-6

[161] Nebeling M, Norrie M C. Context-aware and adaptive web interfaces: A crowdsourcing approach[C]. International Conference on Web Engineering. Springer, Berlin, Heidelberg, 2011: 167-170

[162] Schulte E. Neutral networks of real-world programs and their application to automated software evolution[J]. 2014

[163] Cooper K D, Schielke P J, Subramanian D. Optimizing for reduced code space using genetic algorithms[C]. ACM SIGPLAN Notices. ACM, 1999, 34(7): 1-9

[164] Kebir S, Borne I, Meslati D. A genetic algorithm-based approach for automated refactoring of component-based software[J]. Information and Software Technology, 2017, 88: 17-36

[165] Mkaouer M W, Kessentini M, Cinnéide M Ó, et al. A robust multi-objective approach to balance severity and importance of refactoring opportunities[J]. Empirical Software Engineering, 2017, 22(2): 894-927

[166] Mancoridis S, Mitchell B S, Rorres C, et al. Using automatic clustering to produce high-level system organizations of source code[C]. Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No. 98TB100242). IEEE, 1998: 45-52

[167] Mancoridis S, Mitchell B S, Chen Y, et al. Bunch: A clustering tool for the recovery and maintenance of software system structures[C]. Proceedings IEEE International Conference on Software Maintenance-1999 (ICSM'99).'Software Maintenance for Business Change'(Cat. No. 99CB36360). IEEE, 1999: 50-59

[168] Harman M, Hierons R M, Proctor M. A New Representation And Crossover Operator For Search-based Optimization Of Software Modularization[C]. GECCO. 2002, 2: 1351-1358

[169] Ghannem A, El Boussaidi G, Kessentini M. Model refactoring using interactive genetic algorithm[C]. International Symposium on Search Based Software Engineering. Springer, Berlin, Heidelberg, 2013: 96-110

[170] Jiau H C, Yang F P. Facing up to the inequality of crowdsourced API documentation[J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(1): 1-9

[171] Exton C, Wasala A, Buckley J, et al. Micro crowdsourcing: A new model for software localisation[J]. Localisation Focus, 2009, 8(1): 81-89

[172] Sahraoui H, Valtchev P, Konkobo I, et al. Object identification in legacy code as a grouping problem[C]. Proceedings 26th Annual International Computer Software and Applications. IEEE, 2002: 689-696

[173] Mitchell B S, Mancoridis S. On the automatic modularization of software systems using the bunch tool[J]. IEEE Transactions on Software Engineering, 2006, 32(3): 193-208

[174] Mitchell B S, Mancoridis S. On the automatic modularization of software systems using the bunch tool[J]. IEEE Transactions on Software Engineering, 2006, 32(3): 193-208

[175] Tufano M, Palomba F, Bavota G, et al. When and why your code starts to smell bad[C]. Proceedings of the 37th International Conference on Software Engineering-Volume 1. IEEE Press, 2015: 403-414

[176] Arellano C, Dí az O, Iturrioz J. Crowdsourced web augmentation: A security model[C]. International Conference on Web Information Systems Engineering. Springer, Berlin, Heidelberg, 2010: 294-307

[177] Sharifi M, Fink E, Carbonell J G. Smartnotes: Application of crowdsourcing to the detection of web threats[C]. 2011 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, 2011: 1346-1350

[178] Chilana P K. Supporting users after software deployment through selection-based crowdsourced contextual help[D]. 2013

[179] Chilana P K, Ko A J, Wobbrock J O. LemonAid: selection-based crowdsourced contextual help for web applications[C]. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2012: 1549-1558

[180] Chilana P K, Ko A J, Wobbrock J O, et al. A multi-site field study of crowdsourced contextual help: usage and perspectives of end users and software teams[C]. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2013: 217-226

[181] Leimeister J M, Huber M, Bretschneider U, et al. Leveraging crowdsourcing: activation-supporting components for IT-based ideas competition[J]. Journal of management information systems, 2009, 26(1): 197-224

[182] Chang C E, Chao C, Hsieh S Y, et al. Spmnet: a formal methodology for software management[C]. Proceedings Eighteenth Annual International Computer Software and Applications Conference (COMPSAC 94). IEEE, 1994: 57.

[183] Kapur P, Ngo‐The A, Ruhe G, et al. Optimized staffing for product releases and its application at Chartwell Technology[J]. Journal of Software Maintenance and Evolution: Research and Practice, 2008, 20(5): 365-386

[184] Harman M, Lakhotia K, Singer J, et al. Cloud engineering is search based software engineering too[J]. Journal of Systems and Software, 2013, 86(9): 2225-2241

[185] Madhavji N H, Miranskyy A, Kontogiannis K. Big picture of big data software engineering: with example research challenges[C]. Proceedings of the First International Workshop on BIG Data Software Engineering. IEEE Press, 2015: 11-14

[186] Safdar S A, Lu H, Yue T, et al. Mining cross product line rules with multi-objective search and machine learning[C]. Proceedings of the Genetic and Evolutionary Computation Conference. ACM, 2017: 1319-1326

[187] Araújo A A, Paixão M. Machine learning for user modeling in an interactive genetic algorithm for the next release problem[C]. International Symposium on Search Based Software Engineering. Springer, Cham, 2014: 228-233

[188] Amal B, Kessentini M, Bechikh S, et al. On the use of machine learning and search-based software engineering for ill-defined fitness function: a case study on software refactoring[C]. International Symposium on Search Based Software Engineering. Springer, Cham, 2014: 31-45

[189] Stephenson M, Amarasinghe S, Martin M, et al. Meta optimization: improving compiler heuristics with machine learning[C]. ACM SIGPLAN Notices. ACM, 2003, 38(5): 77-90

[190] Jia Y, Cohen M B, Harman M, et al. Learning combinatorial interaction test generation strategies using hyperheuristic search[C]. Proceedings of the 37th International Conference on Software Engineering-Volume 1. IEEE Press, 2015: 540-550

[191] LaToza T D, Van Der Hoek A. A vision of crowd development[C]. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE, 2015, 2: 563-566

[192] Varshney L R. Participation in crowd systems[C]. 2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2012: 996-1001

[193] Groen E C, Seyff N, Ali R, et al. The crowd in requirements engineering: The landscape and challenges[J]. IEEE software, 2017, 34(2): 44-52

[194] Allahbakhsh M, Benatallah B, Ignjatovic A, et al. Quality Control in Crowdsourcing Systems: Issues and Directions[J]. IEEE Internet Computing, 2013, 17(2):76-81

[195] Li K, Xiao J, Wang Y, et al. Analysis of the key factors for software quality in crowdsourcing development: An empirical study on topcoder. com[C]. 2013 IEEE 37th Annual Computer Software and Applications Conference. IEEE, 2013: 812-817