南京大学

NANJING UNIVERSITY

研 究 生 毕 业 论 文

（申 请 硕 士 学 位）

论 文 题 目 面向 **DevOps** 与微服务的软件团队研究

作 者 姓 名 黄璜

学科、专业名称 软件工程

研 究 方 向 经验软件工程

指 导 教 师 张贺 教授

2021 年 **5** 月 **30** 日

学　　号：**MG1832003**

论文答辩日期：**2021** 年 **05** 月 **26** 日

指 导 教 师：　　（签字）

Study on Software Teams for DevOps and Microservices

by

Huang Huang

Supervised by

Professor He Zhang

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER
in
Software Engineering

Software Institute
Nanjing University

May 30, 2021

学 位 论 文 原 创 性 声 明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不句含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：_____

日期：　　年　　月　　日

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：面向 DevOps 与微服务的软件团队研究

软件工程 专业 2018 级硕士生姓名：黄璜

指导教师（姓名、职称）：张贺 教授

摘　要

开发运维一体化（DevOps）与微服务被认为是解决软件需求快速变更并提供快速可靠的软件开发能力的重要范式。行业报告显示，DevOps 与微服务从出现以来就被一些大型 IT 和互联网公司同时采用。DevOps 通过建立开发与运维团队之间的情感共鸣与跨职能协作来打破部门墙。微服务将整体应用分解为微型服务达到独立与快速交付价值的目的。他们在软件组织中的应用与实施深刻地影响了软件团队这个软件开发基础单元。一些企业应用与实施 DevOps 与微服务时，为了解决软件工程的社会技术（Social-technical）属性带来的沟通交流问题，对软件开发中的小型团队进行了一系列的探索与实践。然而，小型团队可能给组织结构和科学技术带来不良影响，因此小型团队在 DevOps 与微服务环境下的适用条件亟待研究。

通过应用定性与定量方法论、软件工程研究中的 ABC 框架和经验软件工程数据收集技术，本文回顾了近五年发表在三个经验软件工程领域顶级期刊与会议上有关软件团队的研究，构建了一种研究软件团队的迭代式混合方法模型（Iterative Mixed-Method Model for studying Software Teams，*IMMMST*）。本文使用 *IMMMST* 对实践中面向 DevOps 与微服务的软件团队进行了三个阶段的研究。

首先，第一阶段的案例研究使用文档分析、小组访谈和调查问卷收集了微战队这种小型团队的工作形式、工作流程以及战队成员对于微战队的观点。这一阶段的研究从微战队在实践中的类型、关键活动、收益和仍然存在的问题四个角度调查了这种小型团队实践并提出了架构解耦、自组织性和实践指导三个方面的持续改进建议。综合考虑微战队的上下文背景后，本文提出了一个包含组织、部门、技术三个层级五个考虑因素（产品重要性、产品安全敏感

性、部门规模、开发过程、架构类型）的小型团队决策框架（Decision-Making Framework for Small Teams，$DMFST$）。

在第二阶段的民族志研究中通过参与观察和访谈在三家企业中收集了软件团队应用与实施 DevOps 与微服务的情况。这一阶段的研究提炼了 DevOps 与微服务在软件团队日常工作中的实践现状，发现了四个主要问题，即 DevOps 在软件团队间的不完整实施、微服务在软件团队中的滥用、顽固的组织结构和 DevOps 与微服务在日常工作中的弱相关性。本文还以 DevSecOps 为例讨论了 DevOps 相关概念的争论。此外，本文也指出了在工业界进行实践创新的两点挑战。

最后，本文利用对三家公司软件团队的观察结果改进了 $DMFST$，得到了一个三个层级六个考虑因素的改进的小型团队决策框架（Improved Decision-Making Framework for Small Teams，$ImDMFST$）。本文通过基于层次分析法的专家调查对 $ImDMFST$ 进行了评估。组织层级在 $ImDMFST$ 中的权重占比接近 50%。组织特征和架构类型被专家认为是最重要的两个考虑因素，对软件组织决定是否在软件开发中使用小型团队起到了重要作用。

关键词：软件团队；小型团队；DevOps；微服务；经验软件工程

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Study on Software Teams for DevOps and Microservices

SPECIALIZATION: Software Engineering
POSTGRADUATE: Huang Huang
MENTOR: Professor He Zhang

Abstract

DevOps & microservices are acknowledged to be important paradigms to tackle rapid changes in software requirements and provide capabilities for rapid and reliable software development. Industry reports show that DevOps & microservices have been adopted together by some IT giant companies since their emergence. DevOps breaks down the barriers between departments through establishing emotional resonance and cross-stream collaboration between development and operations teams. Monolithic applications are decomposed into some micro services to achieve independent and rapid value delivery. The adoption and implementation of DevOps & microservices in software organizaitons have profoundly affected software teams, which could be considered as the basic units of software development. To solve the problems of communications associated with the social-technical attributes of software engineering, some companies have carried out a series of explorations and practices of small teams in software development while adopting and implementing DevOps & microservices. However, small team may have adverse effects on organizational structure and science & technology. Thus, the applicable conditions of small teams for DevOps and microservice need to be studied.

By applying the mixed-methodology of qualitative and quantitative approaches, the ABC framework in software engineering research, and data collection technologies in empirical software engineering, this thesis reviewed the studies on software teams in the recent five years, which published at the three top journals&conferences in the field of empirical software engineering, futher proposes an Iterative Mixed-Method Model for studying Software Teams ($IMMMST$). Using $IMMMST$, this thesis carried out a three-phase research on software teams for DevOps & microservices in practices.

Firstly, the case study (research phase I) used document analysis, group interviews, and questionnaires to collect the work styles and workflows of fireteam, one of the small-team practices, as well as the opinions of team members on fireteam. Fireteam was investigated from four aspects in practices: the types, key activities, benefits and remaining problems. Three continuous improvement suggestions from three aspects of architecture decoupling, self-organization, and practice guidance were proposed to improve fireteam. Moreover, by comprehensively considering the contexts of fireteam, this thesis proposed a Decision-Making Framework for Small Teams ($DMFST$) with three levels (organizaiton, department and technology) and five consideration factors (product importance, product security sensitivity, department size, development process, architecture type).

Secondly, the ethnographic study (research phase II) used participant observation and interview to distill the state-of-the-adoption and -implementation of DevOps & microservices in the software teams from three companies. Research phase II distilled state-of-the-practice of DevOps & microservices in the daily work of software teams, discovered four major problems, i.e., the incomplete implementation of DevOps among software teams, the abuse of microservices in software teams, stubborn organizational structure and weak correlation between DevOps & microservices in daily work. Furthermore, this thesis used DevSecOps as an example to discuss the controversy of DevOps-related concepts. In addition, two challenges for practical innovation in industry were pointed out in the thesis.

Using the observations of software teams from the three companiese, this thesis improved $DMFST$, proposed an Improved $DMFST$ ($ImDMFST$) with three levels and six considerations. This thesis evaluated $ImDMFST$ through expert surveys, which was based on analytic hierarchy process. The weight of organization level in $ImDMFST$ accounted for about 50%. Experts considered organizational characteristics and architecture types as the most important considerations, which play important roles in determining whether to use small teams in software development.

keywords: Software team, Small team, DevOps, Microservices, Empirical software engineering

目　　录

插图清单

附表清单

第一章　绪论

1.1　研究背景与意义

为了快速交付用户价值以应对多样化的用户需求和激烈的市场竞争，软件组织在不断地改进他们的软件开发过程 [1]。面向对象的开发方法 [2]、面向服务的软件架构 [3]、敏捷方法论 [4] 等技术、实践与文化在软件工程的不同发展阶段被软件组织应用到软件开发之中。近十年来，源于敏捷社区的研发运维一体化（DevOps） [5] 和微服务 [6] 在软件组织探索快速价值交付的过程中受到了从业人员的广泛关注与认可。这两个概念的 Google 搜索指数 [7] 自 2010 年以来一直呈现明显的上升趋势，并且上升趋势具有高度的一致性（如图 1-1）。

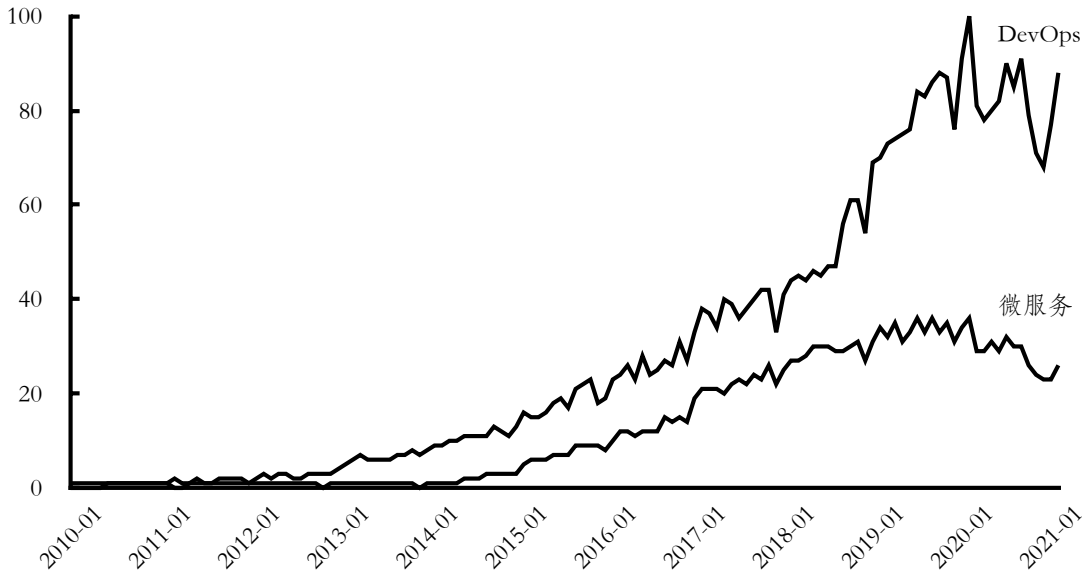


图 1-1: 2010 年以来 DevOps 与微服务的 Google 搜索趋势（按主题）

由开发（*Development*）和运维（*Operation*）组合而成的 DevOps 把敏捷思想和精益方法论拓展到整个软件交付过程，提供了一系列可行可控的软件工程策略、支持了软件设计的扩展、强调了开发与运维团队内外部情感共鸣和跨职能协作，从而保障了价值的快速交付 [5, 8, 9]。DevOps 带来的价值快速交付使其迅速被工业界接受，支持 DevOps 的工具链、实践经验、方法原则和文化理

念等在软件组织中被广泛传播。软件组织期望应用与实施 DevOps 以保障软件开发过程的质量与速度，从而全面提升产品的竞争力。Puppet 实验室的 DevOps 报告曾指出 DevOps 团队的代码生产率和软件稳定性明显高于其他团队 [10]，南京大学发布的 DevOps 中国报告也指出中国软件组织的 DevOps 团队虽然没有达到 Puppet 定义的高性能的标准，但也能够显著地提高团队效能 [11]。

微服务是由微（*Micro*）和服务（*Services*）组合而成。为了实现价值的快速交付，微服务将整体应用分解为通过轻量级消息机制相互通信的微型服务，从而获得可维护性、可重用性、可伸缩性、可用性和自动部署等方面的好处 [12, 13]。作为面向服务的体系架构（Service-Oriented Architecture，*SOA*）的一种演变，微服务在共享组件、服务通信和远程服务访问等方面与 *SOA* 有所区分。此外，微服务的松散耦合使不同的微服务可以使用不同的技术栈进行开发，容器技术的运用让微服务这方面的优势被极大的凸显 [14–16]。

从软件组织的应用与实施上看，DevOps 与微服务是相辅相成的。Puppet 关于 DevOps 的报告指出，在 DevOps 上下文中，微服务与高效的团队软件开发有强相关性 [10]；而 O'Reilly 的微服务报告显示，超过一半使用微服务的公司都在应用 DevOps [17]。一些知名公司正在推广他们针对微服务的 DevOps 解决方案。亚马逊高级专家 David Richardson 等人在一项课程中分享了亚马逊通过 DevOps、微服务和无服务架构对企业快速创新的支持 [18]。微软在 Microsoft Azure 中详细介绍了使用 DevTest 和 DevOps 完成的微服务解决方案 [19]；Google 也在 DevOps 技术指南中对微服务进行了说明 [20]。DevOps 与微服务在企业中的落地也得到了研究人员的广泛关注。他们通过反思在实际企业场景中应用与实施 DevOps 与微服务的案例，得到 DevOps 与微服务的实践经验，期望帮助更多的研究人员和从业人员加深对这两个概念的理解，从而促进 DevOps 与微服务的进一步发展 [21–24]。

软件团队作为软件开发的基础单元，对软件产品具有直接影响 [25]。随着 DevOps 与微服务在软件组织内的应用与实施，软件团队在组织形式与工作内容上发生了变化。DevOps 打破了部门墙，构建了包含开发人员和运维人员的 DevOps 团队 [1]。微服务对整体应用进行了拆分，影响了团队之间的任务分配与沟通 [26]。总的来说，DevOps 与微服务从软件工程师、团队氛围、团队环境和软件制品四个方面为软件团队的实践与研究提供了新的机遇。

小型团队在敏捷软件开发中被认为可以提供有效的团队沟通并激发团队成员的潜力。使用小型团队还可以解决在大型团队中存在的由人与社会方面引起

的亲密度和技能等方面的问题 [27]。在 DevOps 与微服务提供的新机遇中，小型团队被认为可以解决软件开发复杂化过程中沟通、管理等成本激增的问题。一些软件组织已经在小型团队的使用上做出了尝试，如 Spotify 的软件开发模型和汇丰银行的 Pod 团队等。在带来收益的同时，小型团队也可能对企业中的科学技术（Science and Technology）产生破坏 [28]。因此，识别小型团队在软件开发中的使用场景极为重要，在合适的时间对合适的项目使用小型团队进行开发，可以把 DevOps 与微服务带来收益最大化，同时避免小型团队对公司整体组织结构和科学技术的破坏。

软件工程作为一门社会技术（Social-technical）学科，软件团队在其中扮演了及其重要的一环 [29]。软件团队在涉及组织结构与成员关系等社会属性话题的同时，也包含了方法实践的运用、软件代码的产出等技术属性的问题 [30]。作为软件团队的一种类型，小型团队因为规模较小，社会与技术两方面的问题在团队中更加显著，使用单一方法对其进行研究容易遗漏实际应用场景中存在的复杂的组织或者技术环境，而使用混合方法可以通过不同方法的特点来弥补单个研究方法的局限性。这样的 “混合” 可能是在一个范围内使用多种数据收集技术和数据分析方法，也可能是 “混合” 多项研究 [30]。为了研究小型团队的使用场景，需要使用混合方法从多个角度对已有的实践进行调查与分析。这样的混合方法需要包含各种经验研究方法，提供主观与客观、定性与定量以及内部与外部的不同角度的数据 [31, 32]。

1.2 本文主要工作与贡献

本文根据定性与定量方法论 [33]、软件工程研究 ABC 框架 [34] 以及经验软件工程数据收集技术 [35]，回顾了最近五年发表在经验软件工程领域顶级期刊与会议上的软件团队相关研究，总结了一种研究软件团队的迭代式混合方法模型（Iterative Mixed-Method Model for studying Software Teams，$IMMMST$）。基于 $IMMMST$，一个包含三个阶段的研究过程被实施以探究 DevOps 与微服务环境下使用小型团队进行软件开发的过程与经验。

一家大型信息与通信技术企业在他们的软件开发实践中提出了微战队这种小型团队。第一阶段的案例研究围绕微战队展开，使用文档分析、小组访谈与调查问卷的数据收集方法和主题分析与线性回归等定性与定量分析方法收集与分析微战队相关数据。这一阶段的研究从实践现状出发，定义了三种形式的微

战队，识别了团队组建、团队维护、团队内及团队间交流和会议这四个微战队的关键活动。这一阶段的研究还确认了微战队在软件开发实践中的收益。团队沟通与管理成本下降、更高的敏捷性和并发性和个人能力的提升被总结为微战队代码生产率增长的三个主要原因。同时，这一阶段的研究还指出了微战队在实践中的三大问题并提供了三点持续改进方案。结合微战队的上下文环境，一个小型团队决策框架（Decision-Making Framework for Small Teams，*DMFST*）被提出以区分在软件开发中使用小型团队的场景。这一阶段的研究还发现在面向 DevOps 与微服务的软件团队研究中缺乏沉浸式的软件团队视角，这也是开展第二阶段的研究的直接动机之一。

第二阶段研究基于第一阶段研究的发现，旨在从沉浸式的软件团队视角思考 DevOps、微服务，从而发现软件团队和 DevOps 与微服务之间的关系，进一步地探索面向 DevOps 与微服务的小型团队的应用场景。因此，一项跨公司的民族志研究在这一阶段被执行，参与观察和访谈被用于获取 DevOps 与微服务在团队日常工作中的应用与实施情况。对于获取到的定性数据，这一阶段的研究使用扎根理论方法进行分析，发现 DevOps 与微服务在软件团队的日常工作中存在两个主要问题：DevOps 在软件团队中间的不完整实施让一些拥护者宣称的优势无法在软件团队日常工作中得到体现；软件团队盲目追求热门技术和过度追求敏捷性导致的微服务在软件团队中的滥用加剧了微服务带来的一些挑战。在应用与实施 DevOps 与微服务的过程中，顽固的组织结构已经成为了主要障碍，这使得软件组织高层的支持在软件开发中显得尤为重要。这一阶段的研究还发现在软件团队日常工作中，DevOps 与微服务的相关性很弱，只有极少数的从业人员会主动的同时提及这两个概念，开发任务仍然是他们最主要的关心内容，这为 DevOps 与微服务的研究带来了机遇与挑战。

利用第二阶段对三家公司中软件团队的观察改进了 *DMFST*，使之具有更广泛的适用性之后，最后一阶段的专家调查使用层次分析法对改进的小型团队决策框架（Improved Decision-Making Framework for Small Teams，*ImDMFST*）进行了评估并计算了其中各个因素的权重。这一阶段的研究发现，在 *ImDMFST* 中，组织级别的权重值远大于其他两个级别，其中组织特征是权重值最大的考虑因素。技术级别的权重值略高于部门级别，其中架构类型的权重值在六个考虑因素中排名第二。对 *ImDMFST* 的评估打分从专家的角度证实了 DevOps 与微服务应用与实施的主要障碍是顽固的组织结构。

本文的主要贡献有：

1. 总结了一种研究软件团队的迭代式混合方法模型；
2. 描述了一种软件开发中的小型团队实践并提出了三点持续改进建议；
3. 提出、改进并评估了一个小型团队决策框架；
4. 提炼了 DevOps 与微服务在软件团队日常工作中的实践现状与主要障碍。

1.3 文章结构

本文的组织结构如下：

第一章简要介绍了 DevOps 与微服务以及它们给软件团队研究带来的机遇，重点关注了软件开发中的小型团队，强调了在 DevOps 与微服务的环境中，应当使用混合方法对软件团队进行研究。

第二章详细介绍了 DevOps 与微服务、软件团队与小型团队的相关研究。

第三章通过文献回顾，总结了一种研究软件团队的迭代式混合方法模型，并详细描述了使用案例研究、民族志研究和专家调查探索面向 DevOps 与微服务的软件团队的过程。

第四章描述了一种名为微战队的小型团队实践，识别了微战队的三种类型和四个关键活动，针对发现的三个主要问题提出了三点持续改进建议，并提出了一种小型团队决策框架。

第五章探索了 DevOps 与微服务在实际软件团队中的应用与实施情况，DevOps 在软件团队间的不完整实施、微服务在软件团队中的滥用、顽固的组织结构以及 DevOps 与微服务之间的弱相关性被详细地讨论。

第六章使用基于层次分析法的专家调查对小型团队决策框架进行了改进与评估。

第七章简要总结了本文的工作，对未来的工作提出了展望。

第二章　相关工作

2.1　DevOps 与微服务

DevOps 与微服务源自敏捷社区，它们从组织文化、技术手段等多种角度为软件组织快速交付用户价值赋能。近十年来，这两个概念受到了从业人员和研究人员的广泛关注。

DevOps

部门壁垒导致软件开发和运维效率低下，以至于传统软件开发模式越来越难以满足多样化的客户需求和急剧变化的市场 [36, 37]。基于敏捷和精益软件开发的原理，一些从业人员总结了开发和运维过程中的不同观点和思维方式，提出了以自动化基础架构和版本控制为中心的解决方案，并率先引入了 DevOps 来解决开发与运维之间存在的部门壁垒问题 [5]。运动（*Movement*）、实践（*Practices*），文化（*Culture*）和哲学（*Philosophy*）这四个词汇常被拿来修饰 DevOps [38]。荣国平等人则提出了由工具、实践、方法、原则和价值观组成的 DevOps 知识体系（图 2-1） [39]。

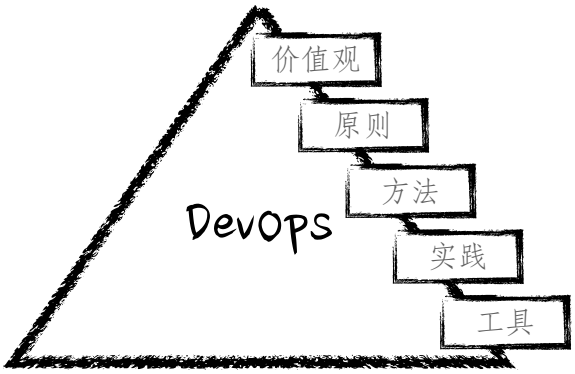


图 2-1: DevOps 知识体系

从组织的角度来看，为了实现快速交付并响应变更，DevOps 强调软件组织中的开发和运维团队内部或团队之间的的情感共鸣和跨功能协作。开发人员和运维人员对交付的产品负有责任，他们需要共同解决遇到的问题。基于

DevOps 知识体系，黄璜等人总结了 DevOps 中的自动化工具面临的三个维度的问题，进一步地提出了软件组织的 DevOps 转型范例 [1]。

从实践的角度来看，DevOps 包含一系列连续的实践（例如，持续集成和连续部署）。戴启铭等人则认为 DevOps 实践可以被分为按阶段划分的阶段性实践和贯穿整体流程的通用实践 [40]。DevOps 通过促进开发和运维团队之间强有力的协作来进行快速开发和部署变更 [41]。DevOps 流水线包含了从版本控制到交付给最终用户的所有软件更改阶段，集成了由开发和运维团队共同或独立完成的 DevOps 实践 [42]。

DevOps 的技术和非技术属性带来了很大的拓展空间。Myrbakken 等人和 Mao 等人发现，DevSecOps 受到了工业界越来越多的关注，软件组织可以从 DevSecOps 中获得诸如安全性之类的收益 [43, 44]。同时，Tomas 等人通过对工具、知识和文化的深入了解，建立了 DevSecOps 三层金字塔模型 [45]。为了增强价值交付并通过自动化弥合要素交付和文档交付之间的差距，Rong 等人提出了一种自动化文档的持续集成方法 DevDocOps [46]。在从业人员与研究人员拓展 DevOps 概念的同时，也存在对 DevOps 相关概念的争论，本文基于研究结果在第 5 .5节中以 DevSecOps 为例进行了讨论。

随着 DevOps 在软件组织中的广泛应用与实施，大量的方法论、实践和工具被提出和传播。现有研究更加关注 DevOps 的总体特征，这导致许多研究人员认为在软件组织中应用与实施 DevOps 仍然是一项艰巨的任务 [47]。探索 DevOps 在软件团队日常工作中的作用将提供新的见解。

微服务

SOA 是为面向服务的计算设计的一种架构风格，为分布式系统的软件应用程序设计提供了范例 [3]。但是二十世纪九十年代后期开发的 SOA 并未为敏捷性和快速部署提供大量支持 [48]。因此，敏捷社区中的开发者们提出了微服务作为解决方案。Martin Fowler 将微服务定义为“将软件应用程序设计为可独立部署的服务套件的一种特殊方式”[6]。作为一组轻量且独立的服务，微服务执行单个功能并通过定义明确的接口与其他服务协作 [49, 50]。图 2-2展示了单体应用与微服务的区别。

微服务的四大典型特征是：围绕业务功能组织系统、自动部署、端点的智能以及对语言和数据的分散控制 [51]。从这四个特征的角度看，微服务可以带来许多收益。Jamshidi 等人证实了 Martin Fowler 总结的三大微服务收益：1）

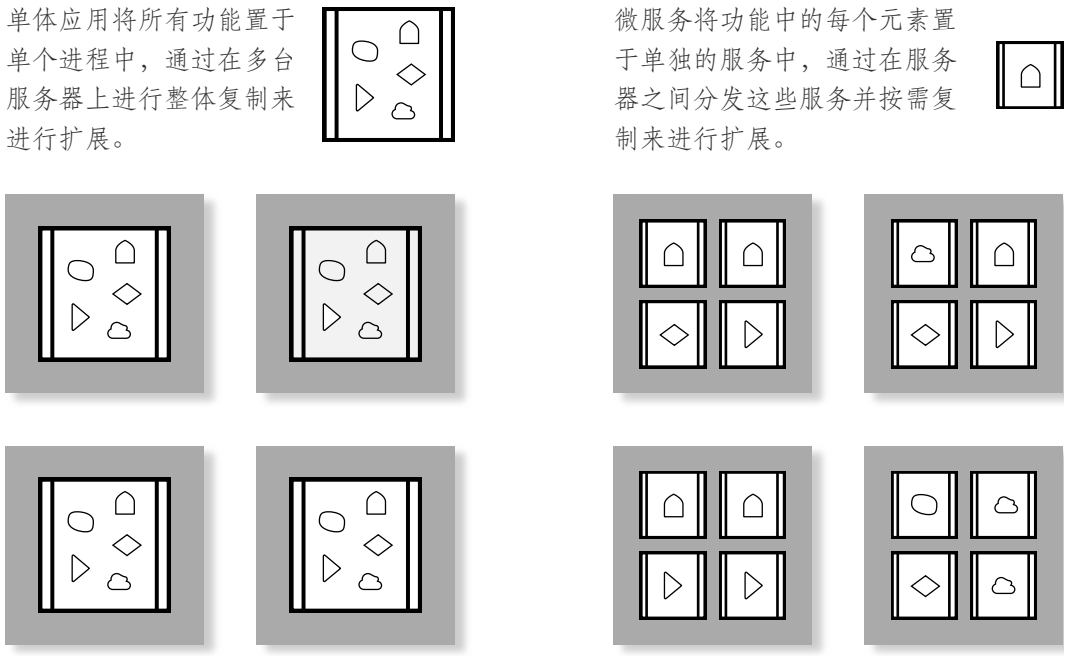


图 2-2: 单体系统与微服务的区别

更快的交付；2）更好的可伸缩性；和 3）更大的自治性 [52]。Soldani 等人则将微服务的主要收益归因于架构的特殊属性、出色的设计模式适用性以及独立部署和管理微服务的可能性 [53]。这些收益让微服务在学术界和工业界得到了越来越多的关注，也让 IT 公司对应用和实施微服务实践充满了动力 [54]。

然而，微服务专注于可以独立部署的单一业务功能，这可能导致产品的组件数量增加，从而增加管理组件的复杂性 [48]。此外，相比单体应用和 SOA，微服务需要更高的网络和计算资源，这可能成为一些公司的负担 [53]。

面向微服务的分解也一直困扰着研究人员和从业人员。旨在为特定问题空间开发解决方案，由 Evans 引入的领域驱动设计是微服务设计中最常见的方法之一 [48, 55, 56]。然而，在设计基于微服务的软件时，确定合适的微服务粒度仍然是主要的难题 [53]。除了领域驱动设计方法外，一些研究人员还在研究使用其他方法来识别微服务（例如数据流驱动方法 [50] 和耦合准则方法 [57]）。

研究人员还关注了微服务的非技术属性。Jamshidi 等人指出组织问题是采用和实施微服务的主要挑战 [52]。Bozan 等人也指出软件组织不应期望现有的根深蒂固的组织结构能够自动适应微服务 [58]。崔海涛等人则认为微服务带来收益的前提是解决组织结构等问题 [26]。因此，对应用与实施微服务的公司的组织文化、发展过程以及团队工作过程进行深入调查将在微服务的进一步发展中发挥关键作用。

DevOps 与微服务的关系

DevOps 与微服务这两个概念自出现以来就紧密相关（如图 1-1中 Google 搜索趋势所示）。微服务通过提高小型团队的重要性来为 DevOps 赋能，一些研究者也因此把微服务视为为 DevOps 和持续交付设计的持续架构 [12, 13, 15]。

基于 47 篇初级研究，Waseem 等人发现 DevOps 中的微服务问题主要可以分为以下三个方面 [59]:

1） DevOps 中微服务的开发与运维，如 Thanh 和 Pahl 等人的研究 [60, 61]。

2） DevOps 对微服务在方法和工具上的支持，如 Miglierina 和 O'Connor 等人的研究 [62, 63]。

3） DevOps 中微服务迁移经验，如 Balalaie 等人的研究 [15, 64]。

图 2-3展示了三个方面中的具体关注的子主题。

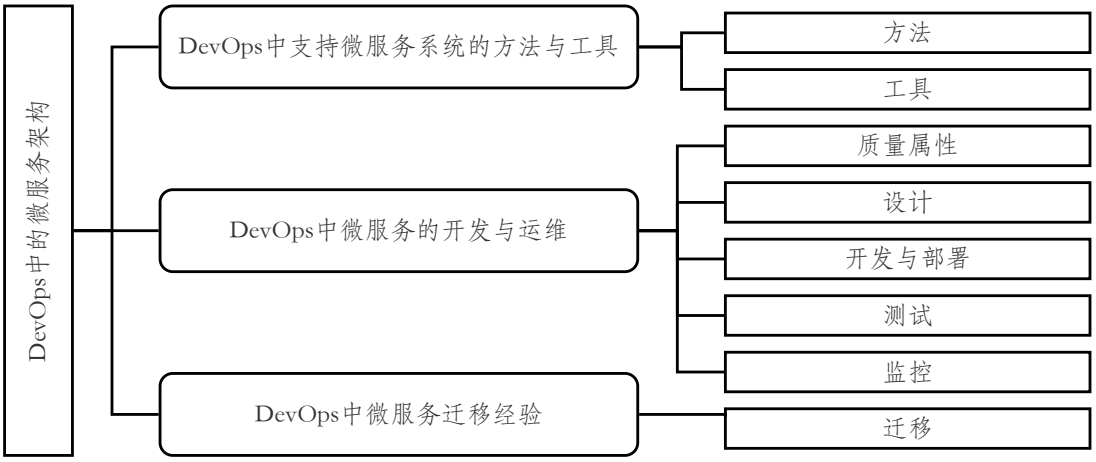


图 2-3: DevOps 中微服务研究主题的分类

Waseem 等人还收集了五十种支持在 DevOps 中开发微服务系统的工具。他们发现大部分研究人员和从业人员认为在 DevOps 中使用微服务对大多数的软件质量属性产生了积极影响 [59]。

Shahin 等人发现以微服务为代表的模块化体系结构与 DevOps 应用与实施的成功最相关 [65]。Choudhry 等人则讨论了物联网、微服务和 DevOps 结合的效果和收益，并建议在物联网的应用与实施中考虑微服务和 DevOps [66]。Rademacher 等人提出了一种基于 DevOps 团队观点的微服务建模语言，并使用了两个案例验证了这种模型驱动方法的适用性 [67]。崔海涛等人的研究则认为 DevOps 是微服务对组织结构产生影响的一个方面 [26]。

对 DevOps 与微服务的研究更多地侧重于技术改进以及它们的总体特征。虽然这些研究对大量的从业人员进行了访谈并观察了一些企业实践，但是它们基于研究人员的先验知识与从软件组织角度的顶层思维，容易忽略实践中有关人和社会方面细微但关键的影响因素。混合方法中可以加入实地研究等民族志方法让研究人员实际参与到软件开发基本单元的软件团队中，并进行参与观察和访谈，这会增加了在研究过程中遇到“重要时刻”的可能性，与案例研究等方法提供的视角一起提供对 DevOps 与微服务的全面的理解 [68]。

2.2 软件团队与小型团队

软件团队在软件工程研究中一直是热点话题，软件团队的研究需要从工程师、团队氛围、软件制品与环境这四个方面（如图 2-4）刻画软件团队的工作细节。

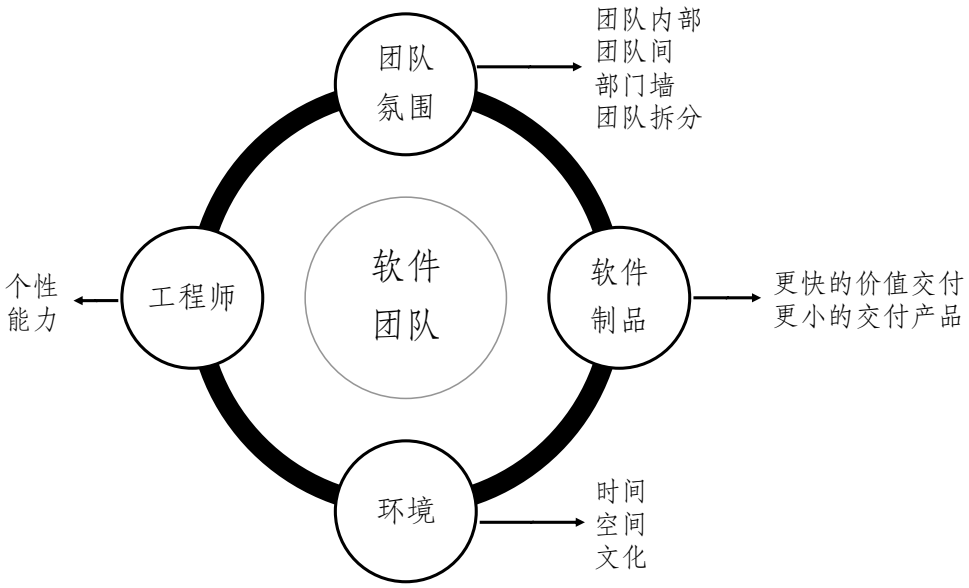


图 2-4: 软件团队研究的相关方面

部分研究者认为“团队 = 软件”，软件团队中工程师的个人行为能够直接影响软件团队的行为，从而与软件团队开发的产品相对应。因此，软件团队的成功需要团队成员之间紧密有效的合作 [25, 69]。如今，用户需求的多样性和市场竞争的激烈性在加剧了软件团队的大型化的同时对软件团队的效率提出了更高的要求 [70, 71]。在现代软件团队中，由于软件团队大型化成为趋势，一些由人和社会方面引起的问题（如友善、才华、技能和沟通能力）逐渐浮现 [72]。软

件组织的管理人员在努力寻求这些问题的有效解决方案。敏捷社区中 DevOps 和微服务的出现，为软件组织解决这些问题提供了新的技术手段和理论依据，面向 DevOps 和微服务的小型团队在实践中不断涌现。这些小型团队被从业人员认为可以表现出更好的团队合作精神，提供更直接和有效的团队内部沟通并充分利用所有团队成员的潜力 [27]。

研究人员对软件团队的研究通常关注敏捷软件团队。与传统的开发过程和实践相比，敏捷方法更加强调团队合作的重要性 [73]。高质量的团队合作可以为敏捷开发中项目的成功做出贡献。虽然一项对 26 家企业进行的调查证实，团队合作质量与团队成功之间没有明显的关系 [74]，但是团队项目的质量在大型和小型项目中是不同的，这其中，团队领导者和团队成员会对团队合作的质量和软件团队的绩效产生不同程度与不同方面的影响 [75]。团队更替在敏捷软件团队研究中也经常出现。大型软件是通过团队合作开发的，而团队的组成可能一直在变化。为了更好地了解这种变化，Hilton 和 Begal 着重研究了软件团队的更替。他们在一些具有高离职率的软件组织中进行了访谈和调查，以探究成员离职和加入的原因。此外，为了控制团队更替的成本和提高团队更替的收益，他们还对来自员工数据库中六年的定量数据进行了进一步分析 [76]。为了了解使用敏捷方法的优点和局限性，Dybå 等人收集了关于敏捷开发的实证研究，提出了人类因素、社会因素和对敏捷方法的理解之间的关系图 [77]。一些软件团队建设准则（如个性和项目对团队建设的重要性）在提高软件团队绩效的同时促进了软件项目的成功 [78]。另外，软件团队建设准则的持续使用与项目成功之间存在显著的相关性。

小型软件团队在敏捷软件开发中也经常被使用。小型项目的敏捷框架（Agile Framework for Small Project，*AFSP*）是一种可适用于多种类型项目的小型软件团队开发框架。该框架以 Scrum 为核心，包含 *AFSP* 流程和 *AFSP* 实践池两个方面，通过产生结构化的过程来为小型团队中的软件开发人员提供敏捷实践的指导。*AFSP* 流程有助于构建正确的软件或系统，*AFSP* 实践池则为正确构建软件或系统提供了技术和工具的支持。此外，*AFSP* 还包含了四种评估手段：风险评估模型（Risk Assessment Model，用于敏捷评估和选择敏捷项目的初始位置）、四维分析工具（4-Dimensional Analytical Tool，用于测量团队敏捷性）、敏捷采用和改进模型（Agile Adoption and Improvement Model，用于评估敏捷实践的水平）和成功的关键因素（Critical Success Factors，用于衡量项目的成功率） [79, 80]。

基于敏捷开发原则，Kniberg 等人首次提出了一个名为 *Spotify Model* 的软件开发模型（图 2-5） [81]。*Spotify Model* 通过跨职能的自组织小队（*Squad*）将 Scrum 和精益方法应用于大型项目 [82, 83]。*Spotify Model* 中的小队有两种主要类型：1）基于产品策略的长期任务小队；和 2）每季度重新协商的短期任务小队。小队队长负责小队之间和小队内部的沟通，以完成需要解决的问题。与此同时，小队成员的工作是在发现问题后找到最佳解决方案。此外，每个小队都有一个产品负责人（*PO*）负责管理工作和匹配待办事项，以及维护代表了软件组织发展方向的高层路线图。在物理空间上距离相近的小队会组成一个软件开发组（*Tribe*），并由一个组长进行领导。在同一个软件开发组的不同小队中有相同技能和任务分工的成员会组成相关的分会（*Chapter*），如测试分会等。最后不同软件开发组中，有着相似的兴趣爱好的成员共同组成一个大的协会（*Guild*），如人工智能协会。

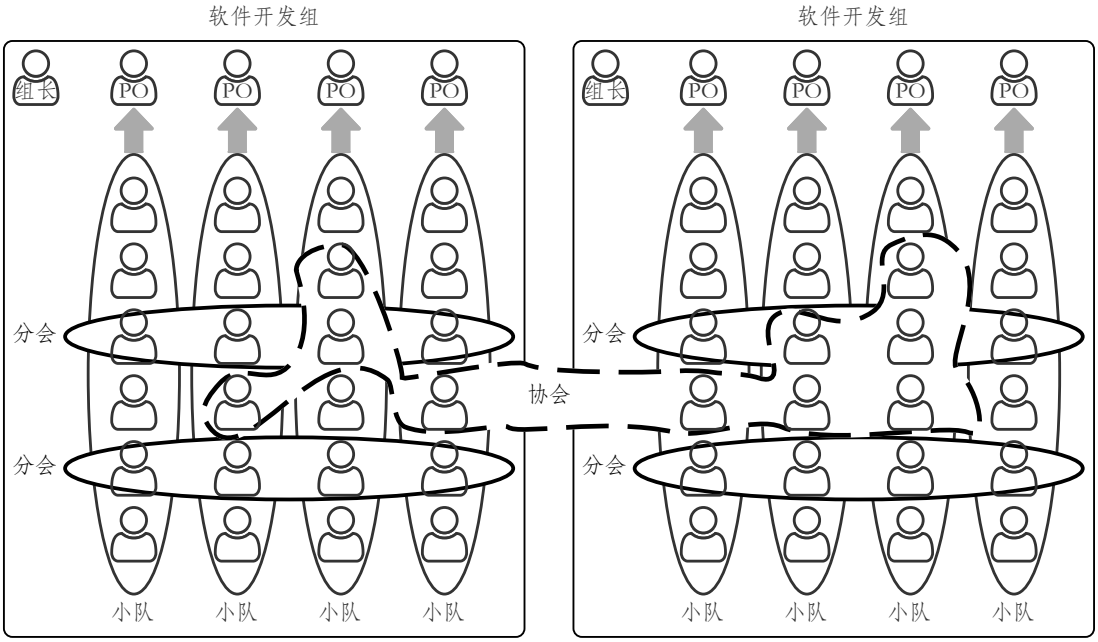


图 2-5: *Spotify Model* 的构成

为了在 B2B 模型的关键任务中提升跨职能自主小队的效率，Salameh 等人对如何调整 *Spotify Model* 进行了为期 21 个月的嵌入式案例研究。通过 14 个半结构化访谈，他们发现了八种可以平衡小队自主权和统一性从而提升小队工作效率的实践。Salameh 等人将这八种实践分为三个类别：1）知识共享；2）基于任务的计划和 3）发布策略 [84]。Salameh 等人还进行了为期两年的嵌入式案例研究，该案例研究基于为期 19 个月的大型离岸任务的案例研究。他们

发现在全球化软件开发实践中，保持 *Spotify Model* 工程文化一致的方法主要有三种，分别是加强集体代码所有权、加强决策和加强团队之间的协作 [82]。*Spotify Model* 中的小型团队解决了软件组织中因为团队大型化而带来的沟通、协作等问题，给软件开发带来了大量的收益。

近年来，随着在 DevOps 与微服务在工业界的应用与实施，培养工程师的能力、团队间的协作与拆分、时间空间对软件团队的影响和更快的价值交付等成为了软件团队研究领域新的关注点，小型团队这种可以提供有效的团队沟通并激发团队成员潜力的团队组织模式也受到了从业人员和研究人员的关注。

2.3 本章小结

本章首先介绍了 DevOps 和微服务以及它们之间的关联，它们在过去的十年中受到了学术界和工业界的广泛关注，为软件团队研究带来了新的机遇。本章还介绍了目前软件团队研究的一些关注点、指出了小型团队在软件开发中的价值并描述了一些敏捷软件开发中的小型团队实践。

第三章　研究方法

软件团队是软件工程研究中的热点话题之一，研究者们使用了各种研究方法对其进行了研究。第2.2节介绍了软件团队研究的四个方面（工程师、团队氛围、软件制品、环境）。这四个方面在 DevOps 与微服务的环境中逐渐复杂化，单一的研究方法不能够全面地描绘它们的准确情况，使用混合方法可以从多个角度对这四个方面进行分析，从而获取更有价值的信息。因此，有效的混合方法论在软件团队的研究中显得尤为重要。本章介绍了一种研究软件团队的迭代式混合方法模型以及基于这个模型设计的三阶段研究过程。

3.1　一种研究软件团队的迭代式混合方法模型

基于 Tashakkori 等人总结的定性与定量方法论 [33]、Stol 等人提出的软件工程研究 ABC 框架 [34] 以及 Lethbridge 等人总结的经验软件工程数据收集技术 [35]，本文回顾了最近五年（2016 年至 2020 年）内发表在经验软件工程期刊（Empirical Software Engineering，$EMSE$）、经验软件工程与度量国际研讨会（International Symposium on Empirical Software Engineering and Measurement，$ESEM$）和国际软件工程评价与评估会议（International Conference on Evaluation and Assessment in Software Engineering，$EASE$）这三个经验软件工程顶级期刊与会议上对软件团队进行研究的论文。

表 3-1: 论文分布情况

	2020	**2019**	**2018**	**2017**	**2016**	**合计**
$EMSE$	6	1	3	5	2	17
$ESEM$	1	2	3	2	4	12
$EASE$	3	1	1	4	2	11
合计	10	4	7	11	8	40

一共 40 篇论文（如表 3-1）被筛选出进行数据抽取，全部论文列表详见附

表 A-1。通过抽取这些论文的研究主题和研究方法，本文总结了一种研究软件团队的迭代式混合方法模型（Iterative Mixed-Method Model for studying Software Teams，*IMMMST*，图 3-1）。

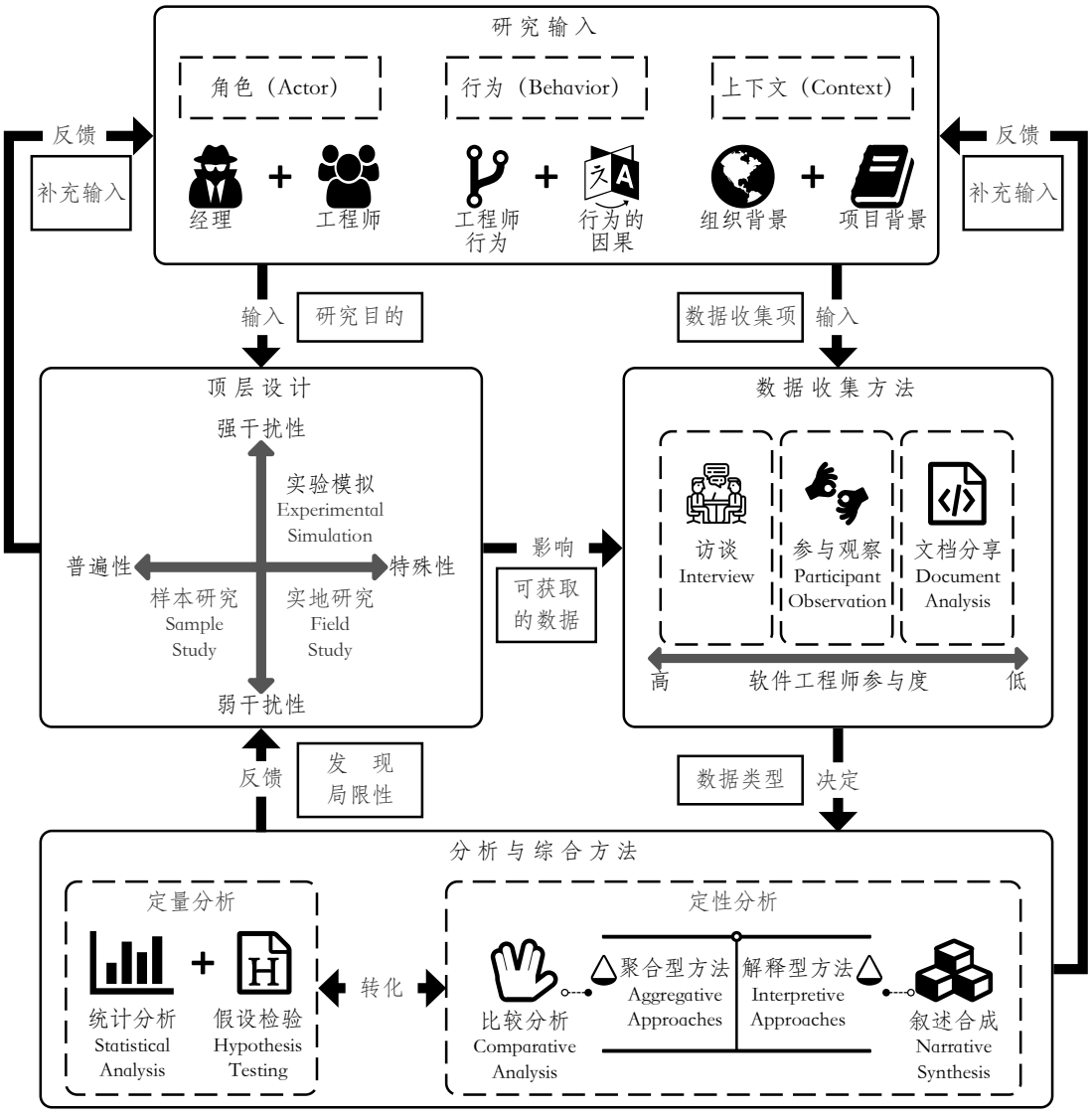


图 3-1: 一种研究软件团队的迭代式混合方法模型 *IMMMST*

研究输入

软件工程研究 ABC 框架指的是在上下文环境下（Context，*C*）探究普遍意义的参与者（Actor，*A*）并精确衡量其行为（Behavior，*B*）[34]。*IMMMST* 中的研究输入通常由这样的角色、行为、上下文构成。其中，软件团队研究中的角色主要包括经理和工程师，在研究一些以用户为中心的实践中，用户也会成为软件团队的参与者（如 [85]）；软件团队研究的行为输入主要包括角色的行为以及这些行为的因果分析，社交闲逛、入职与离职行为、冲突合并和极限编程等团队中的开发与沟通行为最值得关注 [30, 32, 86, 87]；上下文环境通常指软件团队所属企业的背景以及他们研发的项目背景（如 [88, 89]）。

顶层设计

Stol 等人使用研究上下文的普遍性和对研究对象的干扰性来划分软件工程研究方法 [34]。*IMMMST* 以普遍性与干扰性为坐标轴划分了四个象限，软件团队研究的顶层设计坐落于其中三个象限。软件团队的研究大多对研究对象的干扰性都很弱。研究人员为了获取软件团队对于某些实践的真实看法，通常选择案例研究（*Case Study*）作为研究方法。其中，民族志方法中的实地研究（*Field Study*）可以让研究的上下文更加真实，但会让研究更具特殊性 [87, 90–92]；多案例研究（*Multi-case Study*）或者在研究中进行调查取样从而完成一个样本研究（*Sample Study*）则可以在弱干扰性下使研究更具普遍性 [90, 93, 94]。实验模拟（*Experimental Simulation*）可以用来验证团队中某个因素对于团队的影响 [95]，这样的研究对软件团队具有强干扰性且有一定的特殊性，为了控制变量，这里的软件团队通常由高校学生组成。

数据收集方法

Lethbridge 等人认为研究真正的从业人员解决实际问题的方法对于改善软件工程工具和实践十分重要，因此软件工程研究需要现场研究的支撑 [35]。*IMMMST* 中的数据收集方法以 Lethbridge 等人的分类方法为基础，构建了软件工程师参与度由高到低的三类数据收集技术：① 软件工程师直接参与，包含头脑风暴与焦点小组、访谈、问卷、概念建模和工作日志等 [32, 85]。② 软件工程师间接参与，包含参与观察、系统度量和作品等 [87, 96]。③ 研究工作产物，如所进行工作的电子数据库、工具使用日志、文档和系统的静态与动态分析报告等 [97, 98]。*IMMMST* 期望在整体研究中使用多种数据收集方法获取对研究

对象多种角度的描述数据。

分析与综合方法

收集到的数据决定了需要使用怎样的数据分析方法，定性分析方法和定量研究方法都可以被使用。作为一种基于证据的软件工程研究（Evidence-Based Software Engineering，*EBSE*），软件团队研究中存在大量的定性数据，因此，*IMMMST* 中定性分析构成了分析与综合方法的主要部分。定性分析方法可以选择聚合型方法（*Aggregative Approaches*，如分析、元数据概述、内容分析和案例调研），也可以选择解释型方法（*Interpretive Approaches*，如元民族志、扎根理论和叙述合成）[99–101]。在这两种类型的定性分析之外，主题分析同时具有聚合型特征和解释型特征，也经常被用于软件团队的研究。*IMMMST* 中的定量分析主要包括统计分析与假设检验。其中，描述性统计是主要统计分析方法，它通常与定性分析方法一同被使用 [97, 102]。对定性分析方法中一些主题的统计分析可以把研究中的定性数据转化为定量数据，同时，对定量数据的解释与归纳可以把定量数据转化为定性数据。在一些以实验模拟作为顶层设计的软件团队中，假设检验和回归分析等方法也可以被使用 [95]。

迭代过程

*研究输入*是整个研究的开端。对软件团队的研究需要从研究输入的 *ABC* 中获取研究目的，用于指导*顶层设计*，同时还需要获取数据收集项支持*数据收集方法*的使用。在完善*顶层设计*的过程中需要不断地补充输入 *ABC*，使之能够支持整个研究过程，而设计的研究方法从 *ABC* 中获取数据，进而影响*数据收集方法*。*数据收集方法*获取的数据类型决定了使用什么样的*分析与综合方法*。使用*分析与综合方法*得到的发现以及局限性被反馈给*顶层设计*进行下一轮的研究设计，这个过程中可能会发现一些补充输入被反馈给*研究输入*。通过这种迭代式的混合方法对软件团队进行研究，可以获得对研究对象更为全面的认知。

3.2 研究总体设计

DevOps 与微服务的应用与实施给软件团队的组织带来了新的机遇，很多软件组织开始使用小型团队来解决团队工作中由人与社会方面带来的沟通、管理和人际关系等问题。针对软件开发中的小型团队，本文基于 *IMMMST* 设计的了如图 3–2所示的三阶段研究过程。

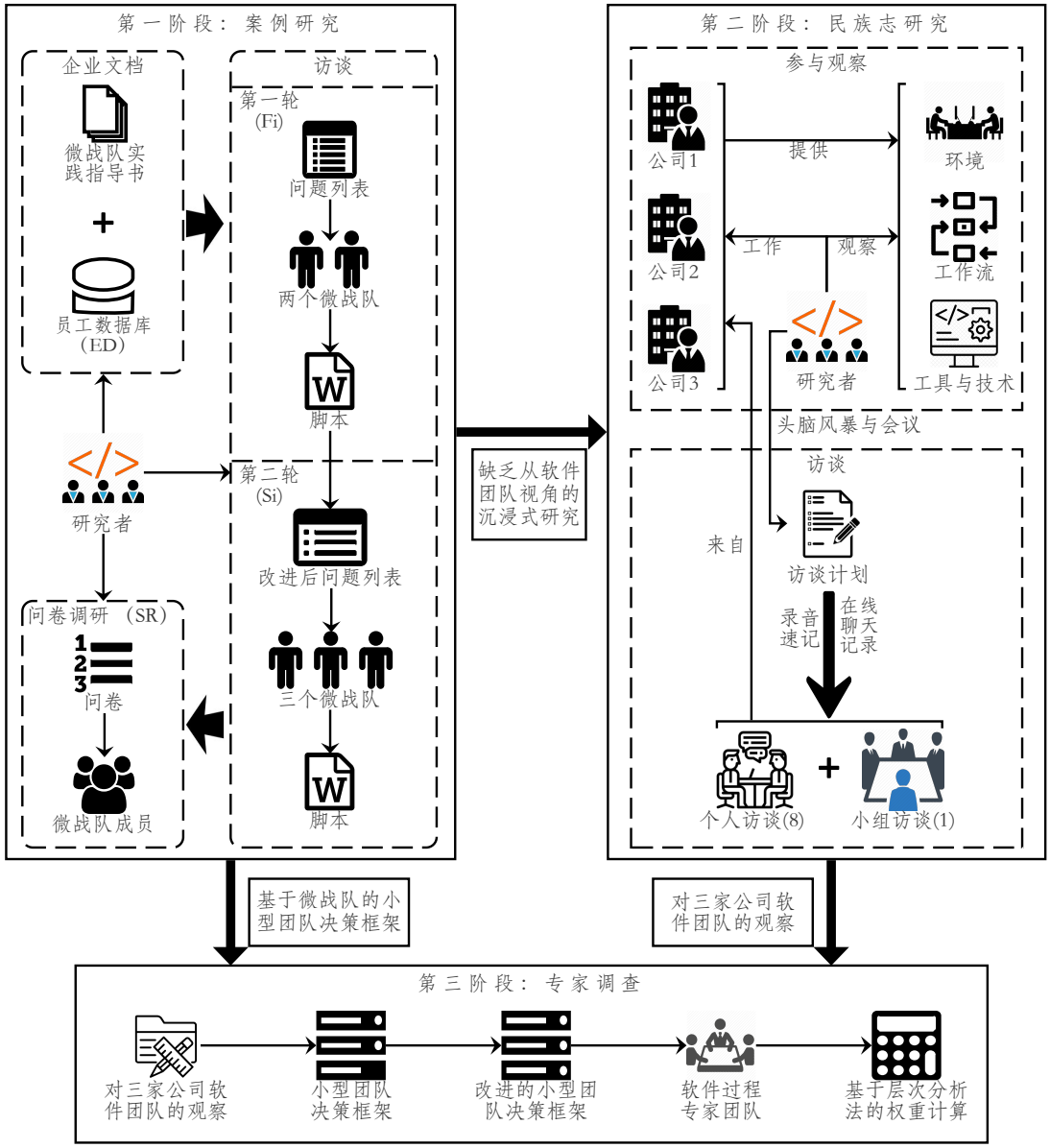


图 3-2: 本文三个阶段的研究过程

中国一家全球化的信息与通信技术企业（表 3-4中的 *C*1）在软件开发实践中定义了微战队这种小型团队来解决传统软件团队中存在的工作分配和协调问题。与 DevOps 及微服务相关的实践在微战队中得到了应用与实施。企业期望微战队这样比传统敏捷团队更小的团队能够让团队成员更加紧密地工作，带来更多的收益，从而减少项目负责人在管理上花费的时间，提高团队的生产力。本文第一阶段的研究以微战队的相关背景（包括微战队的成员、项目以及他们

在实践中的具体做法）作为研究输入。一个案例研究被用来确定组织良好的微战队在软件开发中的作用、制定有据可依的策略来提升微战队在软件开发中的效率以及探究类似微战队的小型团队在软件开发中的应用场景 [90, 103]。多种数据收集方法和数据分析与综合方法被用来从多个维度对微战队进行探索与研究。这一阶段主要围绕以下几个研究问题展开：

研究问题 1-1 微战队在实践中是如何组建与工作的?

研究问题 1-2 微战队给软件开发带来了哪些收益?

研究问题 1-3 微战队在实践中还存在哪些问题?

研究问题 1-4 类似微战队的小型团队适合哪些软件开发场景?

第3.3节介绍了案例研究的研究过程，第四章详细描述了案例研究的结果。第一阶段案例研究的结果中缺乏对 DevOps 与微服务的实践细节及其与软件团队的关系的沉浸式观察，这意味着需要从软件团队和开发者的角度对软件团队实践进行参与观察。同时，对于软件团队和 DevOps 与微服务之间关系的参与观察可以进一步地探索面向 DevOps 与微服务的小型团队应用场景。基于这样的反馈，本文设计了第二阶段研究，即一项包含了参与观察与访谈的跨企业民族志研究 [68, 104]。这一阶段的研究围绕着以下几个研究问题展开：

研究问题 2-1 DevOps 在软件团队中的应用与实施情况是什么样的?

研究问题 2-2 微服务在软件团队中的应用与实施情况是什么样的?

研究问题 2-3 DevOps 与微型服务和软件团队的日常工作会产生哪些相互影响?

第3.4节介绍了民族志研究的研究过程，第五章详细描述了民族志研究的结果。第一阶段案例研究中提出了小型团队决策框架（Decision-Making Framework for Small Teams，$DMFST$），第二阶段民族志研究对三家公司软件团队的进行了参与观察。对不同公司小型团队的观察可以泛化 $DMFST$ 的应用场景，同时引入软件过程改进专家的建议可以为软件组织在软件开发中使用小型团队提供了多个维度的参考。因此本文基于第二阶段的观察结果改进了 $DMFST$，设计了基于层次分析法的专家调查对改进的小型团队决策框架进行了打分评估 [105, 106]。本阶段研究围绕以下问题展开：

研究问题 3 小型团队决策框架中，哪些因素应当被着重考虑?

第3.5节介绍了专家调查的研究过程，第六章详细描述了改进的小型团队决策框架和专家调查的结果。围绕面向 DevOp 与微服务的软件团队，本文关注

软件开发中的小型团队，在研究过程中使用 *IMMMST* 进行迭代演进，最终形成了这样的三阶段研究过程。

3.3 第一阶段：案例研究

案例研究关注现实世界中的现象或案例本身 [90, 92, 103, 107, 108]。这一阶段的案例研究通过**文档分析**、**小组访谈**和**调查问卷**三种数据收集方法收集了微战队的实践信息，其中微战队成员在小组访谈与调查问卷中参与程度高，在文档分析中参与程度低，结合这三者的数据收集可以反应较为全面的实践情况。三种数据收集方法提供了定量和定性两个方面的数据，本阶段使用了混合的数据分析与综合方法对这些数据进行分析。

文档分析

文档分析是一种定性研究形式，研究人员需要对文档中的定性数据进行解释，并围绕主题给出其含义 [34, 109]。两种类型的企业文档提供了对微战队的第一印象：① *微战队实践指导书*，指导书作为开发人员实施微战队的参考文档，其中包括使用微战队的背景、微战队的概念、特性、理论说明和示例，这一类型的企业文档用于获取对微战队的初步了解；② *员工数据库*，提供微战队的信息，例如战队的成立时间、人员组成、平均代码生产率等，这一类型的企业文档用于发现微战队在成立前后表现的差异。为了充分收集每个微战队的信息，成立时间六个月的微战队没有被纳入本文的分析范围。由于保密原则，只有 23 个微战队（表 3-2）的文档数据可以被获取，这些微战队的平均成立时长为 15.6 个月（中位数为 20 个月，标准差为 5.6）。

小组访谈

访谈试图了解人们如何通过言语和行为来思考现象 [92, 110, 111]。两轮小组访谈（ `Fi`和 `Si`）被用来从微战队成员那里收集有关微战队的可靠信息。表 3-3展示了受邀的微战队，被邀请的微战队可以自主选择参与访谈的战队成员，每个战队可以有多个成员参与小组访谈。第一轮访谈（`Fi`）包含了两个微战队，第二轮访谈（`Si`）包含三个微战队。为了便于实地访问，这些负责不同项目的微战队全部来自该企业位于南京的研究所。

`Fi`的问题列表包含被访谈成员背景、被访谈战队背景、微战队工作流程、微战队工作与生活和开放问题五个部分共 18 个问题（详见附录 B.1）。这些问

表 3-2: 微战队的分布情况

城市	可获取文档数据的微战队数量（个）	接受调研的微战队数量（个）	接受调研的微战队成员数量（人）
南京	11	11	17
北京	8	8	10
成都	2	2	4
武汉	1	1	7
西安	1	1	2
上海	0	2	4
合计	23	25	44

表 3-3: 两轮访谈中的受访者

微战队编号	访谈阶段	团队规模	参与访谈人数（人）	项目类型
F1	`Fi`	9	6	网络
F2	`Fi`	6	4	平台
S3	`Si`	4	1	网络
S4	`Si`	4	1	云服务
S5	`Si`	4	3	云服务

题是在微战队实践指导书的基础上，依据以下四个方面的问题扩展而成：1）微战队在实践中的组织形式；2）微战队在实践中的工作形式；3）微战队带来的收益；和 4）微战队在实践中还存的问题。在第二轮访谈之前，`Fi`的访谈记录被初步分析来完善访谈问题。`Fi`的第一部分和第二部分分别有一个问题被删除。由于答案具有相似性，`Fi`的第四和第五部分被合并。最终，`Si`的问题列表包含了三个部分共 16 个问题（详见附录B .2）。

问卷调查

问卷调查通过对受访者对一些问题的回答来评估某种现象在一个群体内的总体想法、观点和感受 [92, 112, 113]。一项针对整个企业（跨多个研究所）的微战队问卷调查被实施来证实访谈得到的发现。包含两个部分共 25 个问题的问卷（详见附录B .3）被该企业的人力资源部门分发到六个研究所的 38 个微战队中。25 个战队的 44 名战队成员回复了问卷（表3-2），回复率是 65.8%。在至

少有一个回复的 25 个战队中，有两个战队没有明确的成立时间，其他 23 个战队的成立时间最短为 3 个月，最长为 22 个月（平均值为 13.7 个月，中位数为 14 个月，标准差为 6.8）。

数据分析

小组访谈提供了大量的定性数据。本文基于访谈记录进行了两次编码迭代：首先，快速浏览访谈记录，标注一些标签以说明对微战队的印象；其次，对有关微战队部分的标签进行聚类。聚类后的标签使用*主题分析* [102] 和*叙述合成* [114] 进行结果的展示。

调查问卷提供了定性和定量数据。*描述性统计*用于展示编码后的定性数据，这部分数据来自 7 个调查问卷中的开放性问题，*统计分析*和*线性回归*用于分析来自 18 个封闭性问题的定量数据之间的相关性。

除了通过《微战队实践指导书》、访谈和调查问卷来探索和确认对于微战队的认知，员工数据库中的企业存档数据（定量数据）也被使用*统计分析*来辅助整体的研究过程。

3.4 第二阶段：民族志研究

民族志研究从宏观和微观角度探索人、组织、文化之间的细节，试图发现环境与现象之间的联系 [68, 92, 115, 116]。软件工程研究中的民族志方法可以用来理解从业人员如何按照流程和方法执行软件开发实践（例如 [117]）、提出或演进软件开发技术（例如 [118]）、探究环境对软件开发中利益相关者的影响（例如 [119]）以及探究团队协作中利益相关者的行为模式（例如 [120]）。民族志方法中包括了大量的数据收集技术 [91, 121]，这阶段民族志研究使用*参与观察*和*访谈*两种数据收集技术来获取软件团队日常工作中 DevOps 与微服务实践的应用与实施情况。参与观察与访谈中，软件团队成员的参与程度都较高，能够提供软件团队和开发者对于软件团队实践的理解。民族志的参与观察和访谈提供了大量的定性数据，本阶段研究使用*扎根理论*方法的开放编码、轴向编码和选择性编码对其进行分析。

进入实地环境

民族志研究需要使用合适的方式进入实地环境 [91]。对三家公司（表 3-4）的参与观察和访谈在 2020 年 1 月至 2021 年 1 月期间分为三个时间段进行。

表 3-4: 三家公司的基本信息

	公司 1（*C*1）	**公司 2**（*C*2）	**公司 3**（*C*3）
类型	电信	大数据与人工智能	软件服务
规模	超过 10 万名员工	约 1000 名员工	约 100 名员工
产品	信息通信基础设施等	数据库与云平台等	工作管理系统
客户	政府、大学、金融机构、能源公司、交通公司，个人用户等	政府、大学、金融机构、能源公司等	大学、餐饮公司、个人用户等
全球化	全球化企业	本地企业，谋求全球化	本地企业
经验	DevOps（5 年）、微服务（6 年）	DevOps（1 年）、微服务（5 年）	DevOps（3 年）、微服务（5 年）
受访者	开发人员、架构师（4 人）	开发人员、（助理）项目经理（2 人）	开发人员/架构师、产品经理、首席技术官

首先，作者 1 月至 4 月在 *C*3 担任实习产品经理，这是一份兼职工作。由于 Covid-19 的爆发，*C*3 提倡在家工作，因此有三个月的参与观察为在线参与观察。此后，作者 7 月至 8 月被 *C*2 聘为全职项目经理助理。2020 年 12 月，作者与一些其他研究人员通过横向科研项目加入 *C*1，观察 DevOps 与微服务相关实践。*C*1 在软件开发实践中提出的微战队在案例研究中被描述，在这一阶段作者进入 *C*1 的横向科研项目与微战队案例研究不同，主要观察的团队与第一阶段有所区分。三家公司中软件团队成员和其他合作员工的日常活动和行为在参与观察期间被重点关注。

参与观察

参与观察是民族志方法中最主要的数据收集方法 [68, 122]。DevOps 与微服务相关实践的环境、工作流程、工具与技术这三个方面在这一阶段的研究中被观察。表 3-5显示了研究中的一些具体的观察项目，符号 “√” 代表在该家公司完成了这个观察项目。

受进入 *C*1 实地环境的方式的限制，对开发环境的观察限于对源代码进行分析，并且 *C*1 的内部会议不允许被观察。这可能对观察结果造成一些不利影响（如对一些玩笑的误解），为了弱化这些影响，一次小组访谈被组织实施。

表 3-5: 三家公司中的观测项

观测项	*C*1	*C*2	*C*3	描述
企业文化	√	√	√	企业文化为熟悉公司和了解某些决定的原因提供了很大的帮助。这一部分重点关注公司历史、宣传手册、培训课程以及员工对公司的认同感。
工作环境	√	√	√	工作环境包括公司的工作时间、工作空间、下午茶和员工护理等。这个观察项还将考虑竞争产品和公司的生活环境。
沟通交流	√	√	√	沟通交流主要出现在调查人员与其他员工之间。观察时，通信工具、内容、频率和效率都受到关注。
现行开发流程	√	√	√	现行开发流程侧重于软件团队中使用的软件开发过程，这些通常在文档中被指定。工程师的行为会被一起考虑。此外，研究人员将实际参与这些过程。
开发环境	×	√	√	开发环境不同于过程，更加关注语言和工具上的细节。此观察项还包括一些开发资源（例如需求文档、UI 设计图、源代码、运维报告等）。
会议	×	√	√	会议包括站立会议、周例会、业务会议和其他一些会议。会议中讨论的主题、会议组织形式和会议参与者的表现会被观察。

访谈

*C*2 和 *C*3 的访谈在作者离职之后进行，受访者来自作者所在的软件团队，*C*1 的访谈与参与观察同时进行，受访者从横向科研项目的合作团队中选取，表 3-4展示了三家公司中的受访者。*C*1 中的小组访谈成员包括不同团队的开发人员和三位架构师，其中开发人员还接受了个人访谈。*C*3 中接受访谈的开发人员还负责一些软件架构的设计工作。由于 *C*3 规模较小，因此首席技术官也以团队成员的身份参与日常开发工作。

为了了解 DevOps 与微服务在软件团队日常工作中的作用，包含职位和主要职责的描述、受访者与 *DevOps* 的故事、受访者与微服务的故事和团队工作的组织这四个主题的访谈计划在遵循 “受访者应详细阐述自己的观点” 原则的前提下被执行。第一个主题在 *C*2 和 *C*3 中用于了解自作者离职以来，这些受访者的职责是否发生了变化，而在 *C*1 中这个主题用于了解不同团队的受访者的基本信息。第二和第三个主题包含受访者在 DevOps 和微服务实践中的经历，包括学习经历、学习动机、日常使用情况等。第四个主题讨论了团队组织，包括 DevOps 与微服务的现有实践对诸如组织结构和团队成员关系之类的非技术问题的影响。访谈中对主题的讨论是开放的。由于受访者的背景并不完全相同，

因此每次访谈可能包含一些与主题相关的特殊问题。例如，“DevOps 对于需求分析有哪些帮助” 在针对项目经理和产品经理的访谈中出现。这些与主题相关的特殊问题属于软件团队的工作范围，能够充分调动受访者的积极性，帮助研究者从舒适的访谈氛围中获得沉浸式的结果。

数据分析

扎根理论是通过语言和交流来构建理论的方法之一 [100, 123]。本文使用的 Straussian 版本扎根理论包括了开放编码、轴向编码和选择性编码三大策略 [123]。

开放编码是从原始数据中识别概念类别，这是扎根理论的第一步。此步骤中生成的编码旨在尽可能保留访谈和参与观察的原有词汇。例如，结合观察到的软件开发工作流程（图 5-2），*C*3 的访谈中“在内部产品反馈群中，软件开发人员使用产品感到不舒服时会要求产品经理为其优化产品设计” 被标记为“*C*3 中的线上交流” 和“产品反馈未集成在 *C*3 中的流水线中”。对 *C*2 开发过程的观察表明，需求管理系统（*JIRA*）和代码存储库（*Gitlab*）在技术上是被关联的，但是软件开发人员实际上并未在日常工作中将需求与他们自己的代码进行关联。这被标记为“*C*2 的开发者在日常工作中没有关联需求与代码”。

轴向编码通过比较和合并在开放编码阶段识别的编码来生成一组新的编码。“产品反馈未集成在 *C*3 中的流水线中” 和“*C*2 的开发者在日常工作中没有关联需求与代码” 被进一步编码为“*DevOps* 流水线中的鸿沟”。

从轴向编码中找到试探性核心之后，通过选择性编码产生遵循这些核心的理论。围绕暂定的核心“*DevOps* 流水线中的鸿沟”，理论“*DevOps* 在软件团队间的实施是不完整的” 逐步浮现。

3.5 第三阶段：专家调查

层次分析法是一种将定性和定量相结合的系统性分析方法，它利用定量信息将实现目标的决策过程数学化，从而找到优先级相关问题中的更重要的指标 [105, 106, 124]。第三阶段研究首先基于对三家公司中软件团队的观察改进了 *DMFST*，然后使用专家调查收集软件工程领域专家对 *ImDMFST* 的看法，通过层次分析法计算 *ImDMFST* 中各个因素的权重。

基于层次分析法的调查

本阶段邀请了七名（曾经）从事软件能力成熟度模型（Capability Maturity Model Integration For Software，*CMMI*）评估工作的专家回答层次分析法的调查问卷（表3-6）。

表 3-6: 参与调查的 *CMMI* 专家

专家编号	研究/工作领域	评估经验 Ex（年）
E1	DevOps、敏捷开发、软件团队管理、软件需求工程	3
E2	经验软件工程、软件工程方法论、定性数据整合	6
E3	DevOps、软件质量与运维、软件日志记录与分析	4
E4	大数据开发、机器学习、自然语言处理	1
E5	经验软件工程、灰色文献、自然语言处理	6
E6	软件架构、微服务评估与拆分、区块链与智能合约	1
E7	软件过程挖掘与建模、软件过程仿真	4

CMMI 可以用于软件过程改进从而提升整个组织的生产力与竞争力。*CMMI* 评估专家具有丰富的软件过程改进方面的经验，能够从第三方的视角看待小型团队在软件开发中的作用。*CMMI* 评估组的视角与第二阶段研究中软件团队的沉浸式视角的结合，从多个角度构建了对软件开发中小型团队的认知。调查围绕 *ImDMFST* 中的六个因素构建了 15 个问题，专家需要对每个问题中的因素一相对因素二重要程度进行打分（1、3、5、7、9、$\frac{1}{3}$、$\frac{1}{5}$、$\frac{1}{7}$、$\frac{1}{9}$），其中，1 表示被比较的两个因素具有相同的重要性，9 表示因素一相对因素二极端重要，$\frac{1}{9}$ 表示因素一相对因素二极端不重要。

层次分析法计算方式

首先依据 *ImDMFST* 中的三个层级构建了层次分析方法中的层次树（图3-3）。决策树中的关键目标为软件开发中使用小型团队的注意事项。

然后通过专家打分建构造虑因素的比较判断矩阵 P。使用规范列平均法对比较判断矩阵 P 进行归一化得到归一矩阵 PW，其中 $pw_{ij} = \frac{p_{ij}}{\sum_{i=1}^{n} p_{ij}}$，最后计算出考虑因素的权重矩阵 W，其中 $w_i = \sum_{j=1}^{n} pw_{ij}$。合并七位专家的权重矩阵后得到 EW，其中 ew_{ij} 表示第 j 位专家对考虑因素打分计算的权重 w_i。

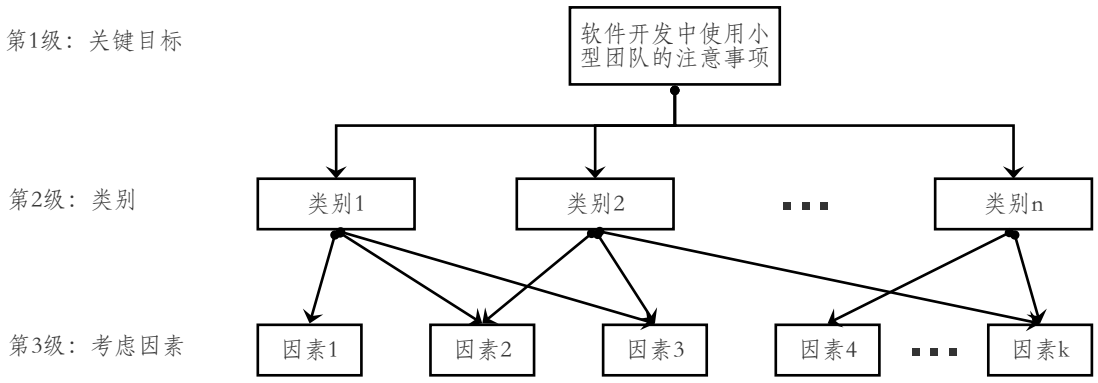


图 3-3: 决策问题的层次结构

$$P=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}, \quad EW=\begin{bmatrix} ew_{11} & ew_{12} & \ldots & ew_{17} \\ ew_{21} & ew_{22} & \ldots & ew_{27} \\ \vdots & \vdots & \vdots & \vdots \\ ew_{n1} & ew_{n2} & \ldots & ew_{n7} \end{bmatrix} \tag{3-1}$$

表 3-7: 不同阶权重矩阵下平均随机一致性指标 *RI* 的值

阶数	1	2	3	4	5	6	7	8
RI	0	0	0.52	0.89	1.12	1.26	1.36	1.41

第三步对专家打分进行一致性检验。通过 n 阶因素权重矩阵计算最大特征值 $\lambda_{\max}$ 和平均一致性指标 CI。通过查阅平均随机一致性指标 RI（表 3-7）计算随机一致性比例 CR，CR 小于等于 0.10 代表了专家打分具有一致性。

$$\lambda_{\max}=\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}\cdot w_j}{n\cdot\sum_{i=1}^{n}w_i}, \quad CI=\frac{\lambda_{\max}-n}{n-1}, \quad CR=\frac{CI}{RI} \tag{3-2}$$

最后进行层次总排序，由第二步中得到的因素权重，综合所有专家的分数与 $CMMI$ 评估经验 Ex（如表 3-6），计算得到第 2 级和第 3 级类别和因素对关键目标的加权相对重要性权值矩阵 GW，其中:

$$gw_i=\frac{\sum_{j=1}^{7}ew_{ij}\cdot Ex_j}{\sum_{j=1}^{7}Ex_j} \tag{3-3}$$

3.6 本章小结

本章介绍了一种用于研究软件团队，包含研究输入、顶层设计、数据收集技术和分析/综合方法的混合方法模型。基于这个模型，本章详细描述了本文设计的案例研究、民族志研究和专家调查的过程。第一阶段的案例研究使用文档分析、访谈和调查问卷多种方法收集数据并应用了主题分析、叙述合成、描述性统计等分析/综合方法对收集到的数据进行分析。第二阶段的民族志研究则通过扎根理论对参与观察和访谈得到的定性数据进行了分析。层次分析法被用于分析第三阶段的专家调查。

第四章　案例研究：软件开发中的小型团队——微战队

*C*1 一直是 DevOps 和微服务的拥趸，很早就在软件开发中应用与实施了 DevOps 与微服务。同时，*C*1 还通过自研的云平台对外提供了 DevOps 与微服务的解决方案，向更多的企业与组织推广 DevOps 与微服务，并助力相关客户企业的发展。表 3-4（*C*1）展示了该公司的一些基本信息。*C*1 庞大的研发团队既带来了好处，也给团队之间的沟通和协作带来了一些麻烦。一些由人和社会方面引起的问题亟待 *C*1 解决（例如，如何有效、高效地管理团队之间与团队内部的沟通）。一些企业管理人员总结提出微战队，并在中国境内的研究所中进行了推广。由于微战队的成员人数远少于普通团队，并且被视为组织中的初级社会组织，*C*1 希望微战队可以使同事之间的工作更加紧密，减少项目负责人的管理时间，从而提高团队生产力。本章详细描述了对微战队的案例研究结果，并进一步地提出了一个基于微战队的小型团队决策框架。

4.1　实践中的微战队

考虑到不同业务交付背景下的不同成员的情况，微战队实践指导书设置了“可独立交付的任务的规模”、“团队人员的技能” 和 “团队对外支持的程度” 这三点微战队需要考虑的因素。在此基础上，为了让微战队能够充分地完成价值交付，指导书还提出了以下五种理想的微战队模式：

外科手术模式：　首席程序员承担微战队的技术职责，其他人员提供配合。

交响乐模式：　微战队成员具有不同的技能，可以互相补充。

学徒模式：　微战队由老员工和新员工组成。

独立研发模式：　研发人员与合作伙伴的员工共同为微战队负责。

客户模式：　微战队中包含客户，客户为微战队提供领域支持。

微战队实践指导书建议每个微战队采用一种理想模式。然而，小组访谈（Fi和Si）和问卷调查（SR）发现，微战队在实践中没有完全参照指导书进

行。一般而言，微战队是敏捷团队的一种变体。微战队可以根据规模以及他们的功能特点分为三种形式：*带扩展的普通敏捷团队*；只负责某一服务生命周期中一个部分的*单一职责团队*；对某个产品全生命周期负责的*全产品周期团队*。

带扩展的普通敏捷团队

这一类微战队可以认为是带有一定扩展的普通敏捷团队，小组访谈中有两个战队被归入这一类型（F1 和 F2），人数分别为 9 人和 6 人。这种微战队包括了敏捷团队需要的人员配置。设计师、开发人员和测试人员在队长（也称为项目负责人）的领导下工作。负责屏蔽外部因素对成员干扰的对外接口人通常由队长担任。这种微战队被看做是微战队的最初形态。

单一职责团队

小组访谈中有一个 4 人战队（S4）被定义为这种微战队，他们只负责一些开发阶段的任务，测试等其他阶段的任务不在他们的职责范围内。这一类微战队的规模比第一类微战队小，而且只承担产品生命周期中一部分任务，所以他们没有明确的职责划分。这类微战队的队长通常也是对外接口人，负责屏蔽外部因素干扰。

全产品周期团队

有两个战队（S3 和 S5）在小组访谈中被识别为这种类型的微战队，他们战队人数都是 4 人。这一类微战队的任务更加细化，负责产品中某个或某几个微服务的整个产品生命周期，开发人员也是运维人员，当服务出现问题的时候，需要及时去修复。

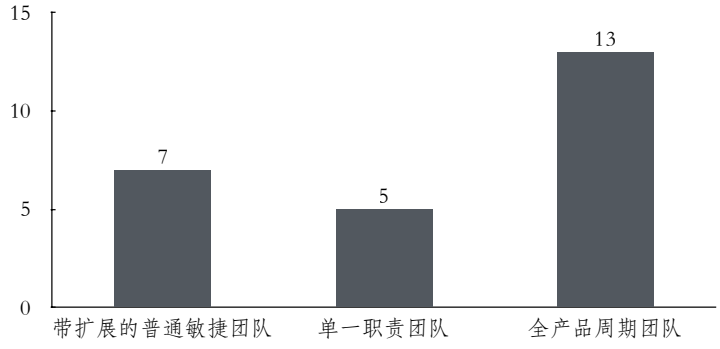


图 4-1: 问卷调查中三种类型的微战队数量

图4-1显示了问卷调查（SR）中三类微战队的分布情况。通过分析他们在

企业中所属的部门，这阶段的研究发现带扩展的普通敏捷团队与嵌入式系统开发更相关，全产品周期团队更可能开发云服务。

4.2 微战队的关键活动

团队组建、团队维护、团队内及团队间交流和团队会议被识别为微战队的四个关键活动。Fi和Si确定了每种活动的影响因素和特殊做法，并通过SR进行了确认。本节详细描述了这四个关键活动并为每一项活动提供了一个场景。

团队组建

团队规模在微战队组建的过程中是极其重要的一个因素，它决定了一个微战队的“微”的程度。关于团队规模，一位受访者称：“我们主要负责两个模块，因此团队规模更大”。这个战队的另一位成员补充说：“团队人数和业务特点也有关系，我们需要和各个产商对接，支撑这一块的人数比较多”。本阶段研究从小组访谈中识别了影响微战队规模的六个主要因素：1) 特性的划分；2) 需求的数量；3) 工作的重要性；4) 组织的安排；5) 成员间关系；和 6) 交付时间。

表 4-1: 影响微战队规模的六个因素直接的相关性

		特性的划分	需求的数量	工作的重要性	组织的安排	成员间关系	交付时间
特性的划分	相关性	1	-.289	-.276	.102	.193	-.352
	显著性		.115	.133	.584	.298	.052
需求的数量	相关性	-.289	1	.343	-.380*	-.392*	.411*
	显著性	.115		.059	.035	.029	.022
工作的重要性	相关性	-.276	.343	1	-.179	-.120	.093
	显著性	.133	.059		.335	.521	.619
组织的安排	相关性	.102	-.380*	-.179	1	.116	-.576**
	显著性	.584	.035	.335		.535	.001
成员间关系	相关性	.193	-.392*	-.120	.116	1	-.164
	显著性	.298	.029	.521	.535		.378
交付时间	相关性	-.352	.411*	.093	-.576**	-.164	1
	显著性	.052	.022	.619	.001	.378	

*. 在 0.05 级别相关性显著；**. 在 0.01 级别相关性显著

这六个因素在问卷调查（SR）中得倒了确认。如果一个回复标记了一个影响因素，那么标记为 1；否则就标记为 0。六个因素之间的相关性在标记完调查问卷后被计算（如表 4-1），其中*工作的重要性*和*需求的数量*在 0.01 层级（双尾）上有显著的相关性，说明了这两个因素在微战队组建的时候总是被一起考虑。与之相反的是，作为反映工程师个人感受的*成员间关系*与其他因素之间并没有显著的相关性。

从这些相关性中可以看出，组织因素（4，6）指导下的技术因素（1，2，3）在微战队组建的过程中比工程师自身感受（5）更重要。这也体现了 *C*1 的公司文化：服从与执行。如某位受访者所说，“*公司的决定执行就行了*”。

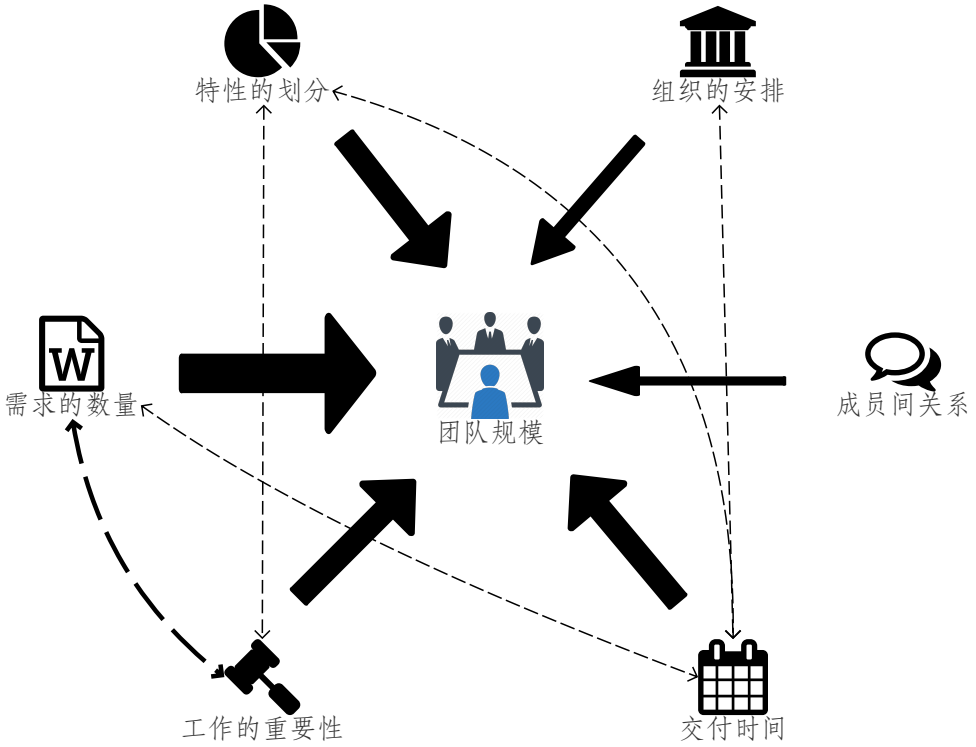


图 4-2: 影响微战队规模的六个因素

图 4-2中，影响因素之间虚线的粗细表明了它们之间的相关性强弱，团队规模和每个影响因素之间箭头的粗细表明了该因素对团队规模影响的强弱。

场景一：为了提高生产力，部门经理决定根据新的微服务架构调整团队安排。最初的团队分为几个较小的微战队。如果一项工作很重要并且有更多需求，那么承担这项工作的微战队的规模将会很大。有时，在组建微战队时，部门经理也会考虑到不同员工之间的关系。

团队维护

团队越小，每一位团队成员在团队中承担的责任越多，成员离职或岗位调整对于团队的影响就会越大。通过有效的手段保证任务在有团队成员离职的情况下持续进行对于小型团队显得更为重要。一位微战队队长在小组访谈中提到，“微战队是有轮换机制的，在同一时间内有至少有两个队长要对某一个服务有了解，.....，当然这种轮换机制中也会有一定的缓冲时间”，另一个微战队的成员也补充到，“不要将鸡蛋放在一个盘子里”。

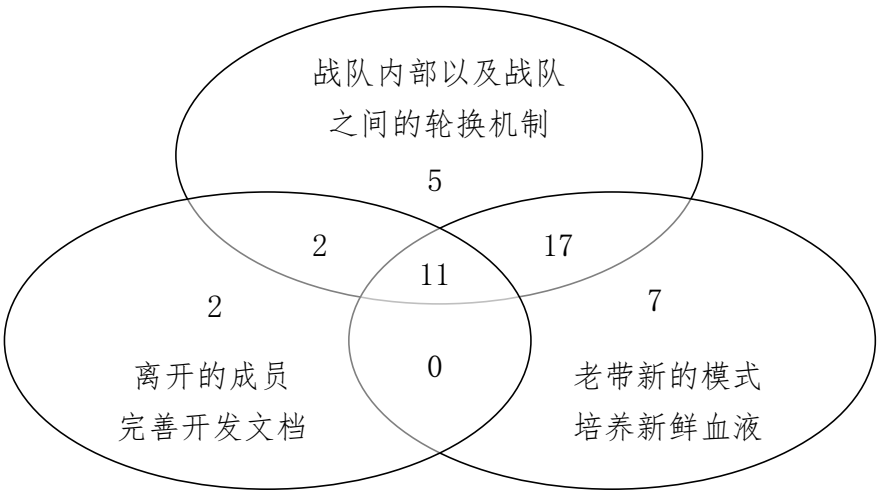


图 4-3: 微战队成员离职的解决方案

这个阶段的小组访谈发现了微战队中的三种应对微战队成员离职的解决方案：1）战队内部以及战队之间的轮换机制；2）离开的成员完善开发文档；3）老带新的模式培养新鲜血液。图4-3展示了在调研问卷中 44 位微战队成员答案分布的韦恩图。分别有 35 位微战队成员认为轮换机制和老带新模式是适合微战队的解决方案，只有 15 位成员认为需要离开的成员完善开发文档。选择轮换机制的同时选择老带新模式的人数达到了 28 人，这说明了两种解决方案具有一定的互补性。考虑到 *C*1 有上万名开发人员，技能备份会是比文档更好的选择，因为他们认为，“技能上的备份让新员工可以更快地融入团队，融入项目中”。

场景 *2*：一名微战队成员因病暂时离开岗位。为了确保团队的正常运作，考虑到工作的重要性，管理层让执行备份的人员在有空的时候接手工作。

团队内及团队间交流

面对面交流和电话会议是最常见的两种交流方式。在这两种方式之外，微战队成员之间的交流或者与其他战队的交流也可能是通过对外接口人进行，这个对外接口人通常由队长担任，比如在Fi中，一位队长描述了他们的交流过程：“我们的队员都在一个地方工作，有问题他们就会面对面的交流，.....，如果涉及其他团队，一般由对外接口人，也就是我，去和他们进行交流沟通”。

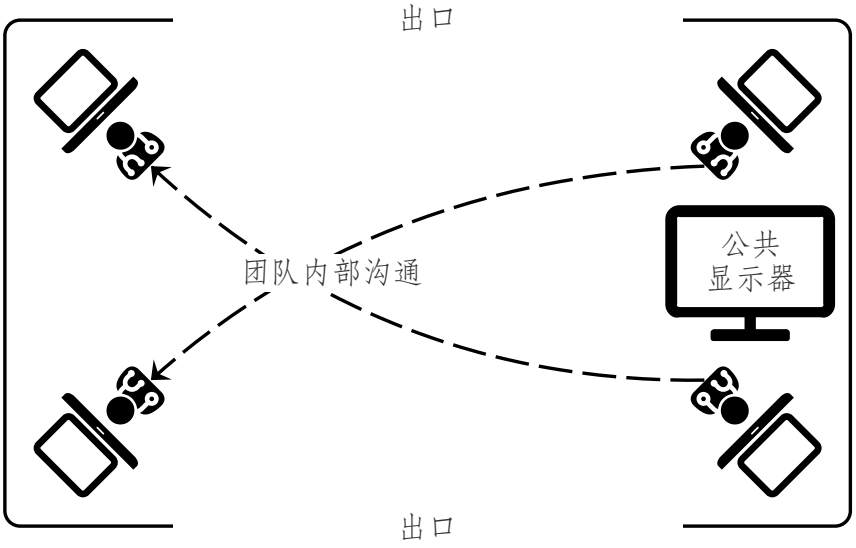


图 4-4: 工作隔间示意图

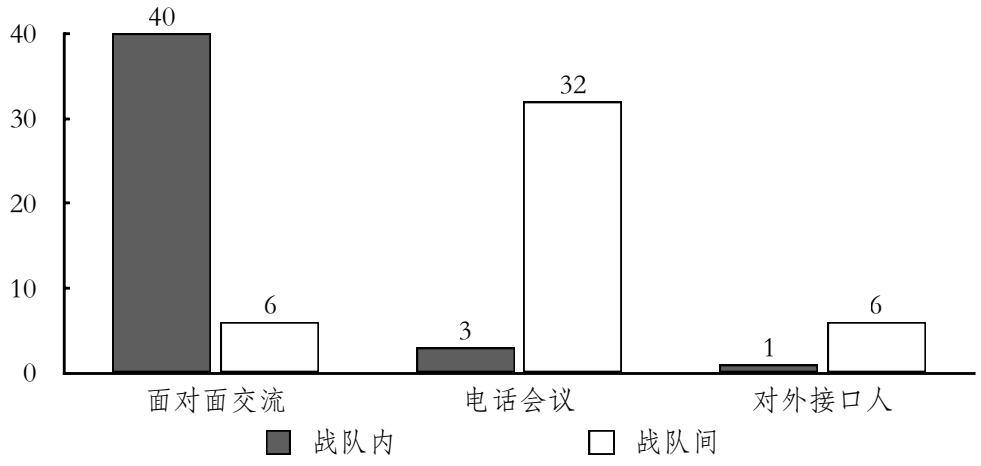


图 4-5: 微战队内部和微战队之间的交流方式

图4-4显示了微战队的一个工作隔间，通常一个微战队的成员在一个隔间或者几个相邻的隔间内，当他们遇到问题的时候可以很快的找到对方进行讨

论。图4-5显示了在调查问卷中发现的微战队内部及其之间的不同沟通方式。大多数微战队成员（40/44）习惯于在战队内部进行面对面交流，而在不同的微战队之间进行电话会议，使用对外接口人进行战队间的交流仍然是少数。

场景 *3*：一个微战队成员需要更改某接口的参数。他发现这是他的微战队编写的界面，因此他拉上维护这个接口的队员一起工作。在工作场所互相讨论后，他们发现修改此接口需要另一个团队的支持，因此他们发起了电话会议来解决问题。如果他们无法立即达成共识，对外接口人会跟进此问题。

团队会议

微战队中灵活的会议模式已经取代了以前的固定会议模式。以晨会为例，一些微战队已经取消了晨会，还有部分微战队使用不同形式简化了晨会的方式，表4-2展现了采用不同方式进行晨会的微战队数量。

表 4-2: 微战队的晨会方式

会议方式	微战队数量	会议方式	微战队数量
没有固定的晨会	11	有晨会，以任务为中心	6
有晨会，以人为中心	7	其他	1

在微战队看来，“现在每个战队人数很少，以前在晨会上展现的东西现在三两句话就可以解决了”。临时会议会被用来处理不同的意见以及统一工作进度，这样的临时会议队长并不一定需要全部参加。微战队的工作间内有公共显示器或者电视 (如图4-4)，需要讨论的问题都会被投影到这样的显示器或者电视上。如果涉及其他战队，每个战队的对外接口人首先会对问题进行讨论。如果问题涉及整体开发方向或者对外接口人没有协商出解决办法，C1 的高层次裁决会议会介入问题的讨论。

场景 *4*：一名战队成员上班后打开 *IDE* 或设计软件，开始努力工作，而无需开会。唯一需要注意的是队长分配的工作。突然，他被要求参加工作会议，但是战队的对外接口人认为他不需要参加这样的会议，因此他无视了这个会议并继续工作。

4.3 微战队的收益

衡量软件团队的生产力是一项复杂而系统的工作，Petersen 等人发现了 16 种反馈性指标以及 22 种预测性指标，Oliveira 等人则发现代码生产率（*LOC*）、时间和工作量是衡量软件团队生产力的指标中最重要的三个 [125–127]。虽然度量软件生产力对于控制和改善软件开发性能存在重大意义，但一些从业者认为应该使用指标来度量项目，而不是开发者和团队 [128]。*C*1 没有将 *LOC* 用作评估团队生产力的指标。由于这个阶段的研究重点在于评估微战队整体的影响，不会对个人生产力进行排名比较，因此从员工数据库中获取的 *LOC* 数据可以用于进行辅助分析。

表 4-3: 微战队的代码生产率 (*LOC*s/月)

战队编号	成立前	成立后	增长率	战队编号	成立前	成立后	增长率
1	1737	1965	13.1%	13	2272	2619	15.3%
2	1153	1544	33.9%	14	1947	2062	5.9%
3	1737	1965	13.1%	15	2272	2619	15.3%
4	903	1531	69.5%	16	1911	2933	53.5%
5	4125	4320	4.7%	17	2272	2619	15.3%
6	2584	4928	90.7%	18	2272	2619	15.3%
7	2392	4213	76.1%	19	1408	1591	13.0%
8	1251	3599	187.7%	20	2272	2619	15.3%
9	2272	2619	15.3%	21	1408	1591	13.0%
10	1273	1326	4.2%	22	1408	1591	13.0%
11	2867	2987	4.2%	23	2272	2619	15.3%
12	2272	2619	15.3%	Average	2012.2	2569.5	27.7%

表4-3展示了 23 个可以获取文档数据（ED）的微战队的代码生产率情况，其中来自同一研究所的一些微战队（例如，第 1 队和第 3 队）在数据库中共享了代码生产率的数据。在成立微战队之后，这些微战队的代码生产率平均提高了 27.7%（增长率的最小值为 4.2%，最大值为 187.7%，中位数是 15.3%）。

在Fi、Si和SR中，团队沟通与管理成本下降、更高的敏捷性和并发性和个人能力的提升被识别为微战队的代码生产率提升的三点原因。

团队沟通与管理成本下降

Fi和Si发现工作进度统一、合理的需求划分、不必要的会议减少以及与客户交流减少是团队沟通与管理成本下降的四个主要因素。

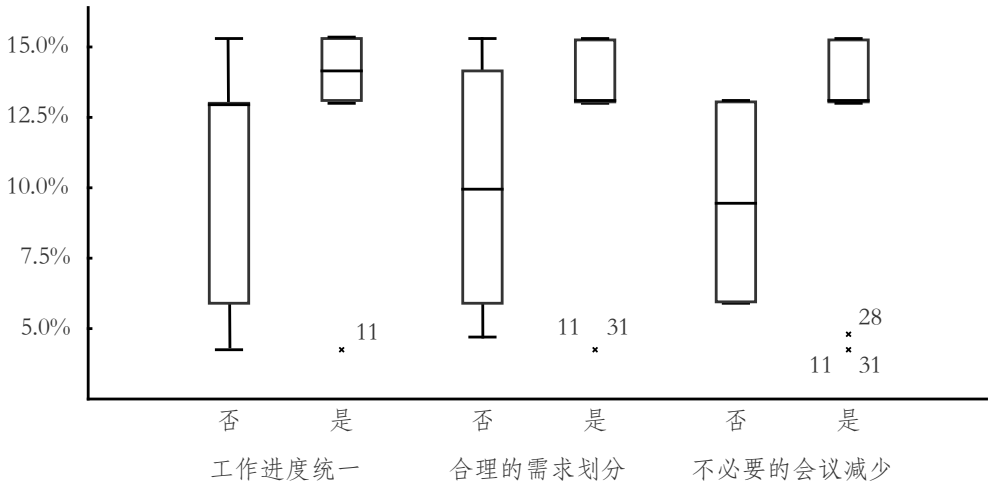


图 4-6: 代码生产率与团队沟通与管理成本下降的三个因素的关系

SR确认了这四个因素。因为不是所有的微战队都需要与客户进行交流，所以“与客户交流的减少”在问卷调查中仅涉及小部分微战队。前三个因素对于代码生产率的增长有着显著的影响 (如图 4-6)，任何一个因素的缺失都会造成生产率的下降 (大约 3%)。同时，由于团队人数减少，团队的管理成本明显下降，SR显示，超过 50% 的队长认为他们仅剩为数不多的管理工作，管理成本下降的一个体现是“他们有更多的时间学习工作外的知识”。

更高的敏捷性和并发性

微战队具有很高的的敏捷性，较小的团队规模让源自敏捷社区的 DevOps 与微服务能够被更加有效的应用与实施。例如，“当很多人一起使用 *DevOps* 流水线的时候，可能别人在用的时候你就用不了，现在使用微服务进行开发，在战队内部使用微战队级别的 *DevOps* 流水线，你就可以随时修改和构建自己战队的微服务”。每一个微战队有自己的流水线（如图 4-7），微战队可以自行控制流水线中的各个阶段，在自己的流水线上进行开发与测试工作，只有可执行版本才允许被提交到企业级流水线上。每个微战队都不需要依赖其他微战队，因为企业级流水线上的版本都是可执行的，微战队的流水线之间也是相互独立的。同时被访谈的战队也指出，在 DevOps 流水线中，因为微服务的应用，看

板这种展示工作状态的方法“在微战队中更加适用，因为微战队包含了产品的大部分生命周期，这是以前的战队没有的”。

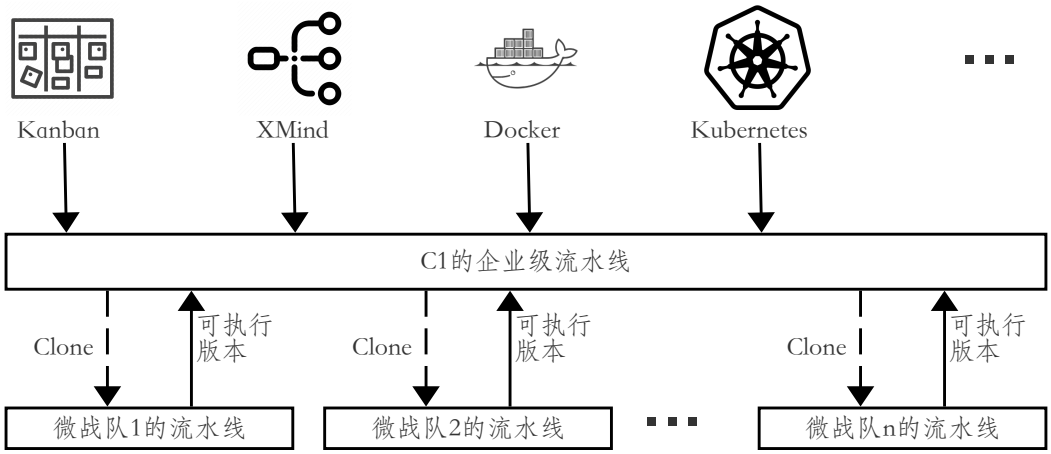


图 4-7: 微战队的工作流水线

个人能力的提升

因为微战队的轮换机制，许多接受访谈的微战队成员认为自己的个人能力有了提升，他们也认为这是自己的代码生产率提升的一个主要原因。从表4-4可以看到，在调研中有 54.5% 的微战队成员同意了这样的观点。这一阶段的研究对战队人数与个人能力提升进行了交叉表的卡方检验，结果显示在 0.01 层级有显著的相关性 (*p-value* = 11.936)，这说明了战队规模与战队成员能力是否提升有紧密的联系。更小的团队中成员需要学习更多的东西，也更容易提升自己的个人能力。个人能力提升的同时，也有一些微战队成员在访谈中提及“他们需要学习更多的内容了，.....，这也让他们的工作压力有所增大”。

表 4-4: 战队规模与个人能力提升

团队规模	没有能力提升	有能力提升	合计
≤ 2	1	0	1
3 ~ 5	8	18	26
6 ~ 8	2	5	7
≥ 8	9	1	10
合计	20	24	44

*C*1 在实践指导书中期望微战队能够实现团队沟通与管理成本下降的目标。在小组访谈和问卷调查中，这一点也被确认为微战队代码生产率增长的主要原因。更高的敏捷性和并发性和个人能力的提升在某些时候可能成为团队沟通与管理成本下降的原因，比如有几位受访者认为“沟通成本下降也有自己能力提升的原因，而 *DevOps* 和微服务实践的应用很大程度上能够帮助团队进行管理，比如可以更清晰地了解工作状态等”。

在为团队沟通与管理成本下降带来积极影响外，更高的敏捷性和并发性和个人能力的提升本身也影响了微战队的代码生产率。DevOps 和微服务早于微战队被该公司应用与实施，微战队的更高的敏捷性和并发性则让这些实践能够被更好的应用和实施。微战队的规模要求战队成员学习更多的知识，能力上的提升也不仅仅局限于解决交流等问题。对于这两个原因，个人能力的提升可以让团队成员能够更好地实施 DevOps 和微服务实践从而进一步提高敏捷性和并发性，而在微战队中采用和实施 DevOps 和微服务相关实践也要求战队成员进一步地提升自己的能力。

4.4 微战队还存在的问题

微战队在为软件开发带来收益的同时，还存在管理和领导力、对微战队理解的分歧和微战队的自组织性这三个问题。本节描述了这三个问题，并简要对比了微战队实践指导书与实践现状。

问题一：管理和领导力

虽然小组访谈（Fi和Si）中有四位微战队队长声称他们不需要在管理上花费很多时间，但一些微战队成员认为他们仍然在管理和领导上花费了大量时间。一名战队成员说，“他（队长）仍然需要给我们分配任务，当我们遇到任何问题时，无论是在开发中还是需要与其他人的联系，我们都将先与他沟通”。图 4-8显示了队长和战队成员对管理和领导成本的不同看法。

问题二：对微战队理解的分歧

研究中微战队成员对微战队的理解不同。在八个超过一人回复问卷调查的微战队中，有七个微战队的成员对三个基本问题给出了不同的答案：

1. 你们微战队中有多少成员？（两个微战队回答不同）

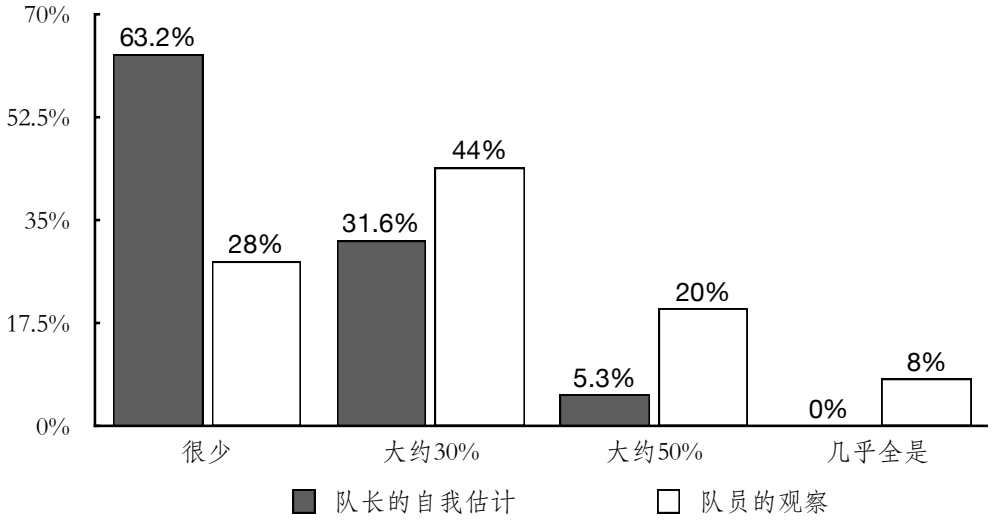


图 4-8: 微战队队长的管理成本

2. 你们微战队有明确的职责划分吗？（三个微战队回答不同）
3. 你们微战队是全功能团队吗？（六个微战队回答不同）

出现这个问题的原因可能是 *C*1 没有为微战队提供全面的培训。一位受访者说，“’我们在成立微战队之前召开了一次会议，告诉我们为什么应该成立微战队，但是没有具体的指导方针”。另一位受访者证实了这一点，“我们仅接受微服务等技术方面的培训，而没有针对微战队的培训”。

问题三：微战队的自组织性

一个团队的自组织性首先体现在团队成员的变更是否由团队自身决定。一位微战队受访者认为“最小的微战队规模应为四人。如果太小，就没有办法应对变化”，另一位受访者则表示“微战队越小越好，较小的微战队表明了更好的架构解耦”。但是，当被问到如何调整微战队规模以及他们是否可以自行调整时，答案是“由管理层安排”。这也反映在技术实践的运用上，虽然微战队拥有自己的流水线（图4-7），但是“工具和流程的使用必须由组织决定”。同样，SR中没有发现太多新的技术实践，这些都说明了微战队的自组织性仍然有待提高。

微战队实践指导书与实践现状

微战队的实践过程与指导书上的期望存在一些差距。接受小组访谈的微战队中有两个微战队的规模大于微战队实践指导书中建议的最大规模（5 人）。

虽然小组访谈的结果给出了一些原因（如这些微战队负责的需求的数量较多、这些微战队与客户交流的较多），但是这些原因与微战队实践指导书提出的实践目标依旧存在差距。所以部分受访者认为一些微战队仍然处于过渡状态，以及他们认为“从软件架构的角度思考，他们开发的产品还存在继续解耦的可能”，解耦以后，过渡状态的微战队可以进一步拆分。

对于微战队的组织形式，*C*1 虽然给出了建议，但是在具体实践过程中，它会受到更多因素的影响，例如，项目的特性、部门领导的性格特征等。微战队被看做是一个初级社会群体，它处于 *C*1 这个更大的社会群体之中，从这个角度看，微战队受到的文化、组织层面的影响也不可忽略。因此在继承 *C*1 团队工作模式的基础上，微战队工作流程只进行了适应性地修改，研究中定义的四个关键过程也并不是微战队所特有的过程实践。

4.5 微战队的持续改进

基于微战队在实践中仍然存在的三个问题，本文从技术、团队和组织三个方面总结了软件架构的进一步解耦、进一步的自组织性和对微战队模式的指导这三点持续改进的建议。

软件架构的进一步解耦

微战队的组建和工作内容与软件架构密切相关，在小组访谈中，大多数微战队成员认为微服务架构的应用是微战队产生的重要原因。架构层面解耦的程度深刻影响着微战队的工作形式与工作效率。

在微战队实践的四个关键过程中，团队组建受技术因素影响最大，而技术因素中，特性的划分如果想要更合理、需求的数量如果需要进一步控制，都离不开对软件架构的进一步解耦。

在小组访谈中涉及架构问题的时候，解耦也是讨论的重点，部分受访者表示，“目前公司架构层面的解耦正在进行，但是还没有达到最好的程度，很多时候在同一个需求上做不到解耦，只能像原来一样进行开发与设计”。

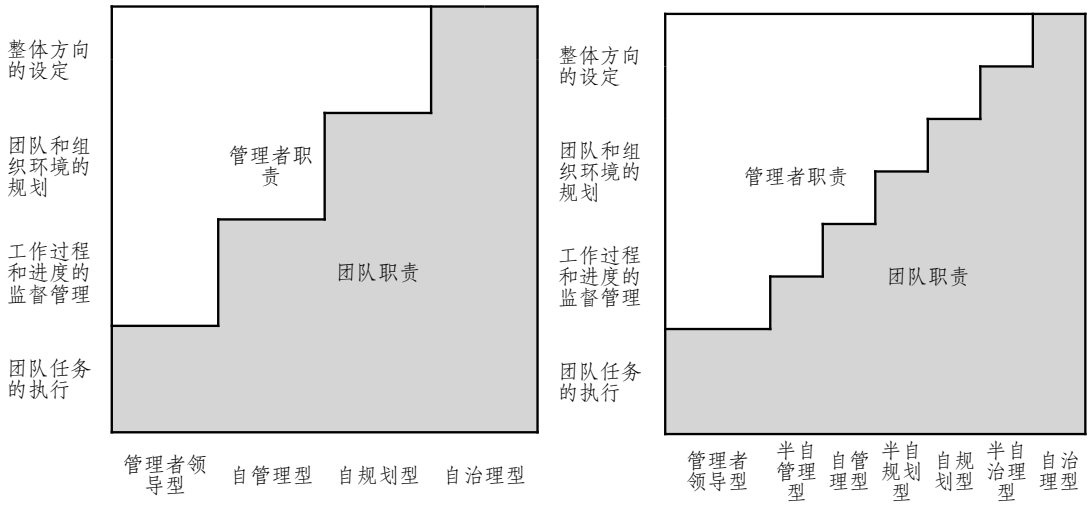
软件架构的进一步解耦是一个循序渐进的过程，它受到组织结构的影响 [26, 129]。康威定律（Conway’s Low）认为一个组织最终产生的设计等同于组织之内、之间的结构 [130, 131]。*C*1 体量较为庞大，软件架构的设计受到其组织结构的影响也是巨大的，这对架构的彻底解耦带来了一些困难。微战队在

软件开发中的持续改进需要进一步解耦软件架构让团队间的关系与微服务间的关系相互兼容。

进一步的自组织性

敏捷宣言中提出最优秀的架构、需求和设计都是出自自组织团队之手。自组织性高的团队往往也能够比其他团队带来更多的好处，比如能够传递更多的商业价值、更加高效地协同工作以及学得更快等。Kirkhope 等人认为无论该决策者或领导人是如何聪明，也无法和整个具有激情的、投入工作的、思考问题的和主动做事的团队相媲美 [132]。Marquet 则认为让团队拥有更高的自组织性可以让企业更偏向于行动派，并让每个人都投入其中 [133]。

微战队的自组织性在该公司内部比较高，但是从小组访谈和问卷调查的结果看，微战队仍属于 Hackman 权力矩阵（如图 4-9(a) [134]）中的自管理型团队。在对权力矩阵进一步细分后，微战队如果能够做到半自治理型团队（图 4-9(b)），那么就可以充分发挥自组织的优势，这个优势是 Keltenecker 和 Hundermark 提出的 “必须让团队竭尽他们所有专业能力，不仅仅是完成他们的工作任务，还要自我监督和控制，自己做决定，甚至设计自己的流程” [135]。



(a) Hackman 权力矩阵　　(b) 改进的权力矩阵

图 4-9: 权力矩阵

小组访谈中讨论到微战队自组织的权力的时候，大多数的微战队成员会认为自己的权力已经足够了。但是处于 Hackman 权力矩阵中第二层级的团队能否发挥出微战队的全部优势值得思考。DevOps 与微服务的应用与实施让越来越多的企业中的软件团队拥有了更多的权力，他们可能也成为第二层级的团队。

在微战队中进一步提高自组织性可以充分发挥微战队规模小的特点，让开放的自我管理激发团队成员的更高的创造力。

对微战队模式的指导

微战队在 *C*1 中缺乏指导。虽然实践指导书对微战队寄予了厚望并且解释了微战队含义，但是缺乏系统的指导和有针对性的实践细节，工程师们很难理解微战队的真正作用。此外，缺乏指导还会造成实践过程中出现与实践指导书不一致甚至相悖的做法，也就很难实现实践指导书所设想的巨大突破。

对微战队模式的指导还可以消除一些对微战队认识的分歧，能够更好地凝聚微战队成员的战斗力，对软件开发工作具有积极的作用。现有针对微服务的指导会涉及一些微战队的内容，但是从小组访谈的结果看，大多数时候微战队成员对于微战队的认知来源于自己的实践。

实践是获得知识来源的途径之一，对于微战队这样的一种新的概念来说，如果想要保证它能够获得理想的成绩，实践之前的指导必不可免，这种指导不应该局限于微战队是什么、为什么以及要怎么做，还应该包括微战队实践指南中所提及的概念以及他们与现有实践的关系。有了这些知识储备，微战队才可以在实践中不断改善。

4.6 一种软件开发中小型团队决策框架

由于微战队是在单个组织的整个企业范围内被制度化的，因此对于微战队的研究被限定于该企业内部。本研究确定了三种类型的微战队，并将每一类微战队与团队规模和团队责任相关联。几名参与研究过程的员工曾指出，原有团队足以完成常规工作，并非每个部门都适合使用小团队进行开发（例如，基础设施部门和安全部门）。

考虑到这些微战队的上下文环境，本文进一步提取了与组织、部门和技术相关的发现，从而提出了包含三个层级五个考虑因素的小型团队决策框架（Decision-Making Framework for Small Teams，*DMFST*）。图 4–10中感叹号表示需要思考的问题，勾号表示小型团队的优势，禁止符号表示不需要使用小型团队。

图 4-10: 一种软件开发中小型团队决策框架 *DMFST*

4.7 本章小结

本章报告了第一阶段的案例研究，通过结合小组访谈、问卷调查和文档分析这三种数据收集技术来探究微战队这种软件开发中的小型团队。

证据表明使用微战队进行软件开发使团队的代码生产率得到了提高，微战队的组建方式以及如何在实践中工作在研究中被调查。三种类型的微战队以及四项关键活动中在研究中得到了确认。微战队代码生产率的提高被归因于团队沟通与管理成本下降、更高的敏捷性和并发性以及个人能力的提升。管理和领导力、对微战队理解的分歧和微战队的自组织性被识别为微战队在实践中面临的三个主要问题。

微战队并没有很好地达到微战队实践指导书对其的预期，本章总结了从技术、团队和组织三个方面对于微战队的持续改进建议，同时也提出了 *DMFST*。

DevOps 与微服务在案例研究中被发现与微战队有着相关性，如更高的敏捷性和并发性让 DevOps 与微服务的实践在微战队中能够被更好的应用与实施。然而，DevOps 与微服务在软件团队中如何为各种利益相关者带来社会技术方面的影响还需要通过沉浸式的参与观察来发现。同时，基于微战队提出的 *DMFST* 是否对于其他企业中的软件团队同样有效仍然需要通过对多个企业的团队进行观察与对专家意见的调查而得到。

第五章　民族志研究：软件团队和 DevOps 与微服务的关系

在软件团队的研究中，沉浸式视角（即软件团队与工程师的日常工作视角）可以为研究者提供对某种软件工程实践的多维理解和更细节的实践信息。第一阶段案例研究缺乏这种沉浸式视角，因此，本文设计了包含参与观察和访谈方法的第二阶段民族志研究。本章描述了民族志研究中发现的软件团队和 DevOps 与微服务的关系。DevOps 与微服务的应用与实施为软件团队带来了一些收益与挑战，对软件团队和 DevOps 与微服务之间相互关系的探索可以为探究面向 DevOps 与微服务的小型团队应用场景提供来自实际软件团队和软件工程师的观点。

5.1　软件团队中的 DevOps

DevOps 为软件团队带来的收益与挑战

DevOps 从软件组织和工程师个人两个角度为软件团队的开发工作带来了收益。

从软件组织的角度来看，DevOps 为软件团队的快速迭代和部署赋能，从而帮助软件组织快速将用户价值推向市场。通过文档化各类规则，软件组织为软件团队建立起基础的软件开发流水线并提供基础的软件开发工具。而软件团队在解决实际问题中会提出的新技术和新方法，并将它们反馈给软件组织。作为一家初创公司，*C*3 使用 DevOps 来实现快速、一致地交付需求，他们每周发布一次或两次更新。在 *C*2 中，超过 70% 的团队每天完成持续集成工作，一些团队平均每天可以运行五次以上的流水线。

从工程师个人的角度来看，DevOps 帮助工程师增强个人能力，在提高软件团队的开发质量与效率的同时，还“可以帮助工程师考虑新的工作机会”。这个好处源于软件组织要求软件团队负责任务在 DevOps 中的整个生命周期。在

DevOps 中团队高频率更替也间接地提高了相关团队成员的个人能力。这个好处一定程度上激发了开发者的学习热情。*C*1 中的一位受访者表示："这对于工程师根据他们的开发任务来更换团队非常有用"。

在软件团队中应用与实施 DevOps 还给团队所属的软件组织和团队中的工程师带来了一些挑战，其中最主要的是高成本。对于软件组织而言，这是采用新技术或新方法时都会出现的问题。*C*3 用了整整一年的时间来替换流水线中 95% 的移动端旧技术。对于工程师个人而言，他们需要承担更多的工作和学习任务。*C*2 中的一位受访者除了负责的开发任务外，还需要负责维护和改进 DevOps 流程。

DevOps 在软件团队间的不完整实施

DevOps 带来的好处和挑战可以在之前的研究中被发现，这表明被观察的三家公司在应用与实施 DevOps 方面属于正常水准。但是，一些诸如"快速发布有助于降低软件团队中工程师的压力水平"之类好处并未被直接观察到，这可能是由 *DevOps 在软件团队间的不完整实施*引起的。

软件组织渴望改进 DevOps 流水线（包括流水线自动化、安全性等），而对 DevOps 提倡的其他方面（例如跨部门沟通）的流程改进则较少，这使得 DevOps 在软件团队间没有得到全面的实施，进一步地让软件团队中的 DevOps 流水线可以被分成彼此无关的几部分（如图 5-1）。

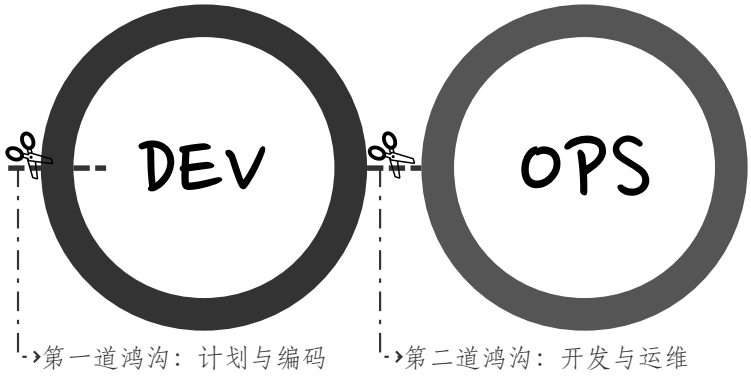


图 5-1: DevOps 流水线中的两道鸿沟

第一道鸿沟在开发（DEV）中的计划与编码之间。虽然 *C*2 在技术上建立了 JIRA 和 Gitlab 之间的关联，但是需求与流水线中代码之间的相关性在实际软件团队的开发中仍然难以捉摸。*C*3 中自行开发的项目管理系统与代码存储库几乎没有关系。另一道鸿沟是运维（OPS）与开发（DEV）。例如，运维问

题只能在 *C*3 的每周运维会议中解决。此外，*C*1 和 *C*2 提供互联网基础设施服务，这一部分的运维与开发完全无关。这种开发和运维的割裂感可能会让人质疑 DevOps 在三家公司的软件团队中是否真正被应用与实施。一些软件团队在日常工作中不需要使用整个 DevOps 流水线，因此，他们只关心自己的团队所涉及的部分，这进一步加剧了 DevOps 的不连续性。在 *C*3 中观察到的工作流（图 5-2）显示了开发活动之间的碎片感。

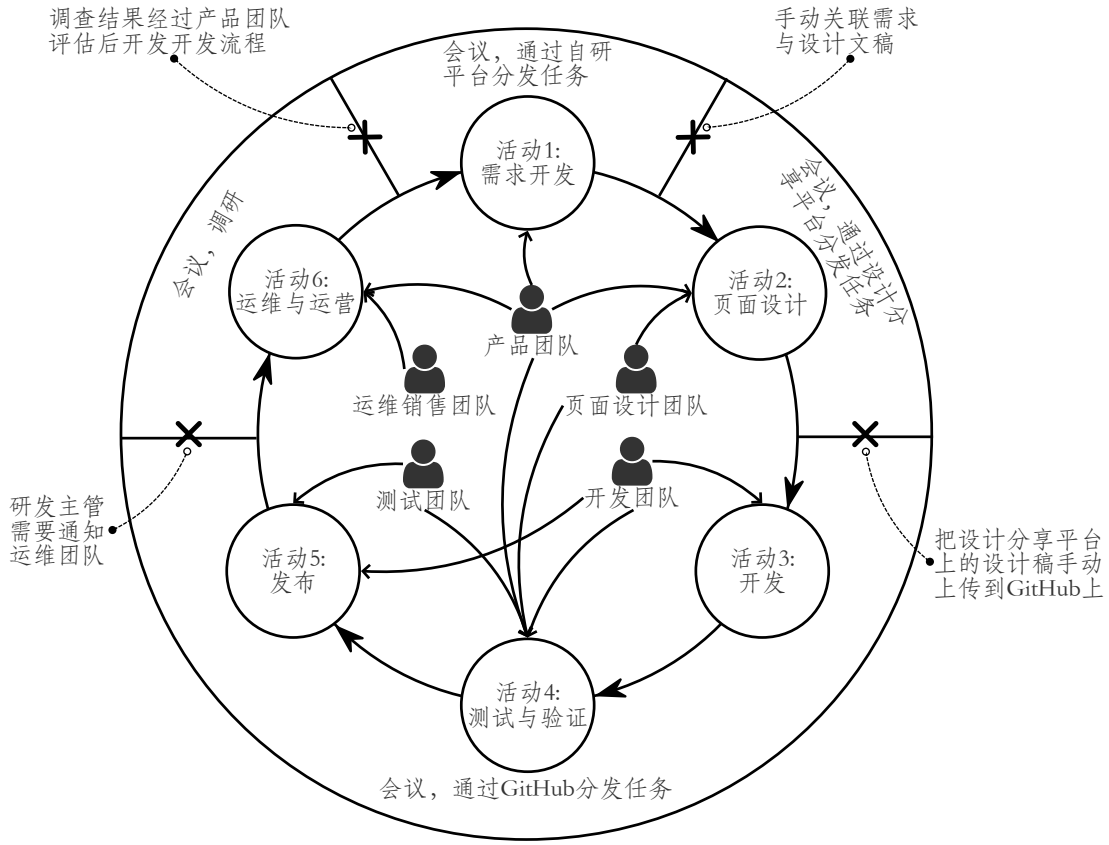


图 5-2: 观察到的 *C*3 工作流

在被观察的三家公司中，DevOps 希望打破的部门墙仍然存在，在某些情况下比以前更难被打破。

DevOps 导致了跨职能任务分配的不确定性，进而使得软件团队在开发时趋向于满足自己的利益。*C*1 中的一位开发人员承认“他希望其他团队复用自己编写代码后自行修改而不是向自己要求更多的功能”。由于四个不同团队的参与，*C*3 中的测试与验证活动（如图 5-2）在整个过程中争议最大，推迟发布新版本软件的原因大多数来自此活动。另一方面，在被观察的三家公司中，虽然存在知识共享平台（例如公开课），但是软件团队更愿意在团队内部而不是整

个公司之间共享技术和方法，这使其他团队的学习过程变得复杂。

这三家公司都声称他们已经在软件团队中应用与实施了 DevOps。但是，观察中三家公司的 DevOps 实践在软件团队中是不完整的，这为 DevOps 成熟度的研究提供了机遇。DevOps 成熟度可以使软件团队对软件组织的 DevOps 实践水平有更清晰的了解，有助于软件组织中 DevOps 的进一步改进。

5.2 软件团队中的微服务

微服务为软件团队开发工作带来的挑战

微服务不是解决架构问题的银弹。对三家公司软件团队采用与实施微服务的参与观察与访谈发现了微服务带来的很多挑战。负责 *C*1 中公司与大学合作项目的软件团队认为，“他们的系统在设计上使用微服务的原因是对敏捷性的关注，他们忽视了的一些业务特点导致了一些架构异味的出现”。*C*3 采用和实施微服务架构是由前任首席架构师决定的，微服务在现有阶段足以满足公司的发展需求，但是他们认为“微服务不会是架构发展的终点”，他们已经开始考虑其他架构设计（例如，无服务架构）。

微服务为软件团队开发工作带来的挑战主要来自两个来源：1）微服务分解和 2）高成本。

虽然研究者们已经提出了许多用于微服务分解的方法（例如，领域驱动设计和基于数据流的设计），但是软件团队有时仍会在日常工作中感到不知所措。这种无力感来自于业务，在 *C*1 中，随着业务的发展，许多微服务开始重组为较大的服务，有些已经不能称为微服务。*C*3 表明，当产生新业务时，划分新领域或使用旧领域既困难又复杂。

高成本主要是由于微服务在软件团队开发工作中需要更多的资源。在 *C*2 的业务会议中经常提到硬件资源的短缺，特别是软件测试的短缺。在 *C*1 中，部署成本不断增加，因为“他们没有部署设计”。此外，微服务中的问题定位还带来了高昂的时间成本，其中“在解决问题时需要确保数据一致性”在 *C*2 的工程师看来花费了他们很多的时间。

微服务在软件团队中的滥用

微服务在软件团队中的滥用被发现是微服务给软件开发工作带来挑战的根本原因之一，其中软件团队“盲目追求热门技术”和“过度追求敏捷性”起到了

推动作用。图5-3显示了一些利益相关者对于在软件团队开发工作中使用微服务的一些看法。

图 5-3: 工作者开发者对于微服务的看法

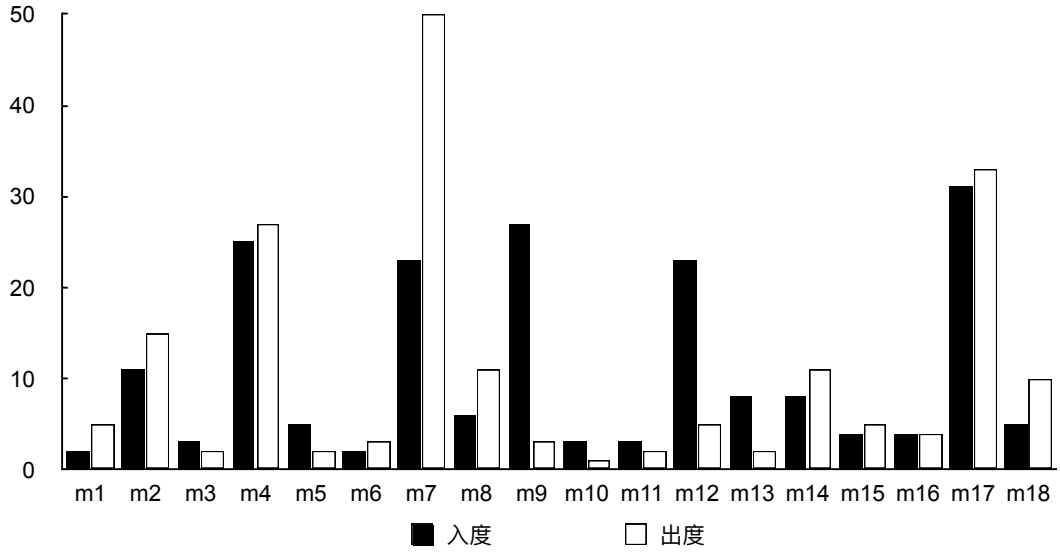


图 5-4: *C*1 中部分微服务的依赖情况（入度与出度）

微服务可能并不适合所有情况。*C*1 的架构师坦白说："目前在电信环境中采用和实施微服务并不是他们的最好选择"。图5-4显示了在 *C*1 中观察到的项目中某些微服务的依赖属性，其中 *m*4、*m*7 和 *m*17 的入度和出度都显著高于其他微服务，这是因为架构师和开发人员在实现软件系统的过程中自发地合并了过度分解的微服务。*C*2 中的工程师"更喜欢 *SOA*，而不是公司声称的微服

务”。*C*3 声称“微服务现在可以支持其业务发展，但是当需求增加时，可能会出现许多问题”。对于软件组织或软件团队来说，尝试新技术是一件好事，但是忽略业务现实，可能会适得其反。

虽然在软件团队中存在滥用微服务的趋势，但我们也应该看到微服务给软件团队开发工作带来的收益（例如，高可靠性和快速迭代）。与 DevOps 一样，软件团队会在使用软件组织提供的工具（例如云和容器）期间提供有关新技术和新方法的反馈。此外，微服务允许开发人员专注于某些特定领域，这减轻了开发人员的负担并增加了软件团队的幸福感。这一点好处还被归因于团队成员数量的减少和灵活的工作环境。

5.3 软件团队组织形式

第5.1节和5.2节描述了软件团队的日常工作中的 DevOps 和微服务。本节详细介绍了参与观察和访谈中软件团队的组织以及它们如何影响 DevOps 与微服务的应用与实施。

顽固的组织结构

三家公司的软件团队组织可以抽象为一个三级结构（如图5-5），技术驱动的变革和文化推广在这样的组织结构中存在一些挑战。

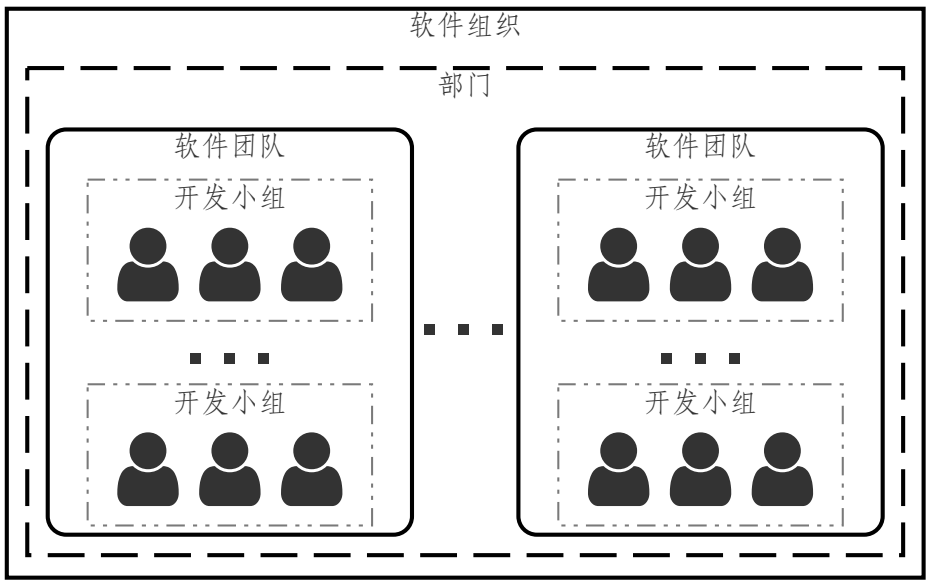


图 5-5: 三家公司中的三级组织结构

技术驱动的组织变革困难重重

组织结构通常由公司的高层管理人员确定，并在正式的组织文档中说明。这里的管理人员“通常是不懂技术的”。另外，管理人员还受到人力资源、财务和其他部门的约束。当部门希望对软件团队进行变革时，决策者将考虑所有团队成员以及其他部门的想法和利益。这个过程持续时间太长，很容易被拒绝，甚至在 *C*1 中，“许多部门负责人都没有魄力去说服决策者”。在图5-5中，技术驱动的创新需要经历从开发小组、团队、部门到组织的四个层次的链条。在部门和组织这两个较高级别的环节中，“技术通常不是解决问题的首选”。

技术文化的传播任重道远

信任是 DevOps 和微服务实践中需要培育的文化核心之一。在这三家公司的跨职能协作中，团队间的抱怨仍然是最突出的协作问题。而且，一些开发人员因为担心承担更多的任务而抵制这些技术文化的传播。此外，“开发人员仍然以他们过去的开发思维方式进行开发”，这种思维方式将需要很长的时间才能改变。虽然在第5.1节中提到知识共享已经成为了这三家公司的共识，但具体如何实施仍需要在软件团队中进一步细化。例如，在 *C*2 的一次有关数据库技术的知识共享课程后，只有不到 10％的参与者对内容发表了自己的看法（图5-6），这表明当前的知识共享氛围仍然不乐观。

负责人：对昨晚的课程进行一次简单的调查。
负责人：如果听懂了50％以上，请回复1。
（无人回复）
负责人：有人懂了20％以上吗？可以发表下自己的意见。
开发人员1：听懂了，但是有点消化不良，总共有90页。
开发人员2：昨晚时间有点太长了。我大概到第80页ppt开始走神。
项目经理1：表情符号[太难了]。
负责人：做[产品XX]的同学呢？你们应该懂的多一些。
开发人员2：也可能从第70页。内容大概是一个磁盘page是一个内存cacheline大小的地方。
项目经理2：表情符号[优秀]。
测试人员1：大差不差能听懂，但忘的太快，得多看几遍才能完全消化。

图 5-6: *C*2 中一次知识分享课程后的对话

顽固的组织结构带来的问题

DevOps 与组织结构的相互关系与影响在第5.1节中已经被讨论。与 DevOps 相比，微服务的技术属性更加突出。作为一种软件架构风格，微服务架构也适用于康威定律（Conway’s Low）所认为的“组织最终产生的设计等同于组

织之内、之间的结构” [130]。除了在第5.2节中讨论的内容之外，微服务还受到了顽固的组织结构以下的一些不利影响。

首先，顽固的组织结构给微服务的开发和维护带来了困难。通常，“微服务的设计需要符合部门之间的预定义规定”，其中工作量和团队利益等问题将被优先考虑。将微服务分配给多个团队，这些团队可能在开发过程中相互竞争（尤其是来自不同部门的团队）。

其次，顽固的组织结构使软件团队的自治性降低。自治性是微服务的支持者认为的优势之一 [136]。虽然软件团队可以在实施微服务的过程中为组织带来新的技术和方法，但是这种软件团队的自治仅限于部门级别（图5-5）。

最后，顽固的组织结构会放大微服务中的通信问题。随着微服务团队规模的缩小，软件团队内部的通信成本降低。但是，微服务团队之间的通信成本急剧上升。在没有良好的组织结构来支持撑这种团队间沟通的时候，这个问题尤为严重。

顽固的组织结构正成为采用和实施新技术的主要障碍。但是，作为软件组织中基本的开发单元，软件团队和开发人员很难解决该问题。因此，软件组织的高层管理人员和其他部门的支持在 DevOps 与微服务的应用与实施中显得尤为重要。

5.4 软件团队中 DevOps 与微服务的弱关联性

第2.1节介绍了 Waseem 等人对 DevOps 与微服务的研究 [59]。他们总结了三类研究主题：① *DevOps* 中微服务的开发与运维、② *DevOps* 中支持微服务系统的方法与工具以及③ *DevOps* 中微服务迁移经验。研究主题①在观察和访谈中被发现，而后两项没有显著的相关性（如图5-7）。

没有受访者主动提及 DevOps 与微服务在日常工作中的关联。对于这三家公司的许多开发人员而言，微服务和 DevOps 之间的连接仅限于“他们对于微服务的开发是在 *DevOps* 管道上运行的”。DevOps 平台可以在有或没有微服务的情况下开发，与此同时，微服务也可以在 DevOps 流水线之外运行，并用其他方式部署和发布。软件团队和开发人员更关心完成日常工作并按时交付项目，而不是关注这些技术之间的联系，“这些只有可能在知识共享课程上提到”。

从软件开发人员的角度来看，所有方法和工具以及经验都应该为产品开发服务。因此，在软件团队的日常工作中，可以将 Waseem 等人提出的三类研究

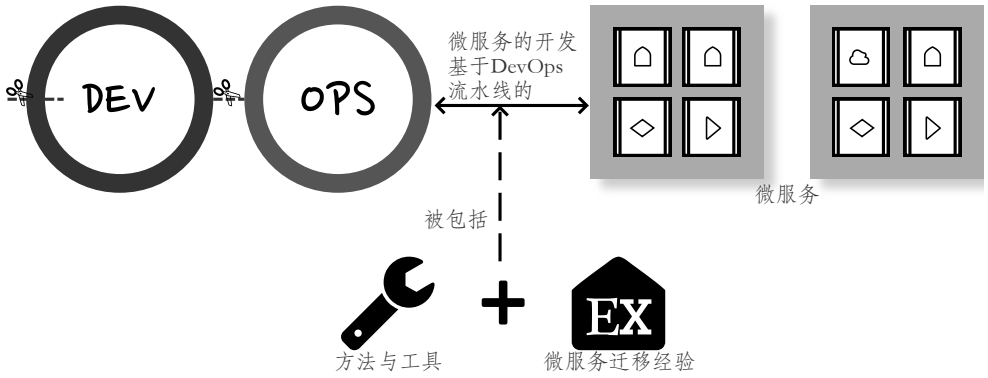


图 5-7: DevOps 与微服务在软件开发日常工作中的关系

主题集成为 DevOps 中微服务的开发与运维这一个主题。例如，*C*2 和 *C*3 使用 Docker 来“部署微服务，以便可以在一定程度上保证 *DevOps* 管道中开发和产品环境的一致性”，这被开发人员认为是属于开发与运维而不是方法与工具。

对 DevOps 与微服务的联合培训可能是软件团队中二者相关性较弱的原因之一。三家公司都为其软件团队提供了技术培训（例如，与 *C*1 与大学的合作课程和 *C*2 在企业内部建立的“公司大学”）。此外，开发人员对特定技术的培训课程（例如 Kafka）和经验案例的培训课程（例如证券行业的创新 AI 解决方案）更感兴趣。

另一个与 DevOps 与微服务弱相关性的重要原因是意见领袖缺乏足够的知识。团队负责人（可以被看作软件团队的意见领袖）影响了其他团队成员对实践以及团队开发过程的理解。尽管他们可能会参加一些实践课程，但他们对实践的知识是有限的。例如，*C*3 的一位负责人认识到“他对 *DevOps* 了解并不是很多”。在这种情况下，研究人员的参与观察可能会触发软件团队对概念的全面认识，从而改进软件团队中的相关实践。例如，安全性已经成为 *C*2 中微服务与 DevOps 之间的桥梁。另一方面，研究人员对于企业实践的研究成果对于实践的进一步发展也至关重要，例如 Waseem 等人提及的监视、安全和性能下降问题可能成为企业在以后应用与实施 DevOps 与微服务时关注的内容。

5.5 DevOps 相关概念的争论：以 DevSecOps 为例

DevOps 衍生出了很多新的概念，如 DevSecOps [40, 44]、DevDocOps [137, 138]、DevOpsRET [139] 等。这一阶段的参与观察发现这些新概念并没有被软

件团队所关注，这也反应了在目前 DevOps 领域中对于一些相关概念的争论。

以学术界较为关注的 DevSecOps 为例，“*DevSecOps*” 作为一种新的概念，强调需要在 DevOps 计划中建立安全基础，期望引起 DevOps 从业人员对于安全问题的思考。自从 Neil MacDonald 在 2012 年首次提出它以来，就被一些从业者认为是 DevOps 的必需的一环 [45, 140–142]。

虽然 “*DevSecOps 是 DevOps 的一种扩展*” 这个概念得倒了普遍的认可，但是仍然有一些从业人员不赞成这个概念。亚马逊的高级解决方案架构师 Margo Cronin 声称 “*安全是 DevOps 的内置属性*” [143]。还有一些其他从业人员支持安全是 DevOps 的自然组成部分，并认为 “DevSecOps” 是多余的术语，如 Mozilla 的安全工程师 Julien Vehent 和 Datadog 的高级工程师 Ilan Rabinovich [144, 145]。此外，一些从业人员意识到在 DevOps 中包含安全的重要性，认为 DevSecOps 属于这个概念的初期，并质疑使用新的缩略语来描述开发、运维和安全协同工作的必要性 [146]。

除了在术语上的争议意外，DevSecOps 中的 “*安全左移*” 和 “*共享责任*” 并不是软件工程领域中全新的主题。Gary McGraw 在本世纪初就提出了 “*在软件生命周期的早期考虑安全性*” 以及 “ *软件安全是每一个人的责任*”，他提供的贯穿整个传统瀑布模型的安全实践仍然是当前安全实践的重要参考 [147, 148] 。

与传统的开发模型相比，DevOps 可实现连续且频繁的部署和交付，但是它带来了更多的安全风险，因此无论是否使用如 DevSecOps 这样的新术语，都需要高效地、自动化地将安全实践嵌入 DevOps 中。

5.6 工业界的实践创新

DevOps 和微服务为实践创新带来了新的机遇。案例研究中讨论的微战队作为面向 DevOps 与微服务的一项团队实践创新，并没有达到 *C*1 的期望。虽然微战队带来了一些收益，但是仍然存在一些问题（例如，管理和领导力），并且在短期内没有被很好地解决。这不仅是由 *C*1 中存在的组织壁垒导致的，在工业界进行实践创新的一些挑战也给微战队以及软件团队带来了诸多挑战。

缺乏知识和培训

缺乏知识和培训将成为企业进行实践创新的一个障碍。虽然企业希望迅速实现商业价值，但在微战队发展的初期，却缺少指导性的理论支撑，从而导

致缺乏必要的实践培训。敏捷实践在企业中落地也曾遇到过相关的问题：由于培训不足，一些组织尚未考虑已经在业界大规模应用的敏捷方法 [149]。在 DevOps 和微服务的出现带来了很多与之相关的实践，组织在这些实践的基础上也会提出新的实践，培训可以作为这些实践在软件团队中快速实施的开端。

根深蒂固的企业文化

根深蒂固的企业文化可能会成为另一个障碍。第5.3节描述了顽固的组织结构对于 DevOps 和微服务应用与实施带来的影响，其中重要的一环就是技术文化的传播。实践的创新和改进与文化息息相关，当试图改变团队之间的协作方式时，微战队遇到了许多困难（请参见第4.4节）。在另一项案例研究中，文化被一些首席架构师认为是采用 DevOps 的重要但是困难的部分 [150]。

虽然在企业环境中进行实践创新可能会遇到许多问题，但是由于不断的改进，值得并鼓励在软件行业中对实践进行创新。

5.7 本章小结

本章报告了第二阶段的民族志研究，调查了实际软件团队中 DevOps 与微服务的应用与实施情况。数据来自对三家公司从文化、工作环境、沟通、现行开发过程、开发环境和会议六个观测项进行的参与观察以及包含了 11 位受访者的 8 次个人访谈与 1 次小组访谈。

三家公司中 DevOps 与微服务的应用与实施（例如收益和挑战）与先前研究中报道的行业实践具有一致性。DevOps 在软件团队间的不完整实施、微服务在软件团队中的滥用以及公司文化中顽固的组织结构是沉浸式观察发现的三个主要现象。这一章还讨论了 DevOps 与微服务在软件团队和开发人员中的弱相关性、DevOps 与其相关概念的争论以及在工业界进行实践创新所面临的两点挑战。

这一章通过对 DevOps 与微服务在软件团队日常工作中的沉浸式研究，理解了 DevOps 与微服务和软件团队之间的相互关系，有助于从概念、实践和可感知性方面了解 DevOps 与微服务在实际软件团队中的实施情况。同时对于三家公司中软件团队的观察，为改进第一阶段研究中提出的 *DMFST* 提供了依据，为第三阶段的专家调查提供了输入。

第六章　专家调查：小型团队决策框架的改进与评估

第一阶段案例研究基于微战提出了小型团队决策框架（Decision-Making Framework forSmall Teams，*DMFST*），第二阶段民族志研究对三家公司中的软件团队进行了参与观察。三家公司的软件开发实践中都存在小型团队。*C*1 提出的微战队在第四章进行了详细的讨论；在 *C*2 中常见一种由一名项目经理与两到三名开发人员组成的产品概念团队（Proof of Concept，*PoC*），他们在项目初期构建产品框架并在与客户的沟通中进行产品概念的快速迭代；*C*3 在软件开发中实施产品经理负责制，现有业务框架内的新需求通常由一名产品经理发起，从开发、测试团队中分别抽调一到两名开发人员组成三到五人的小型团队，产品经理需要追踪需求的全流程开发与运维（如图 5-2）。使用民族志研究中的观察结果对 *DMFST* 进行改进，并引入专家的建议对改进的 *DMFST* 进行评估，可以泛化小型团队决策框架的应用场景，为软件组织在软件开发中使用小型团队提供多个维度的参考。本章详细描述了改进的小型团队决策框架（Improved Decision-Making Framework for Small Teams，*ImDMFST*）以及专家对 *ImDMFST* 的评估打分情况。

6.1　改进的小型团队决策框架

这阶段研究通过对三家公司中软件团队的观察完善了 *DMFST*，进而提出了一种改进的小型团队决策框架（图 6-1）。

组织级别

ImDMFST 新增了组织特征这个维度。三家被观察的公司在组织规模和发展阶段上有着显著差异，因此他们对于小型团队的使用有着各自独到的看法。*C*3 的首席技术官就认为小型团队对于小型企业，尤其是初创企业的软件开发有较深的影响。小型企业中组织架构的灵活变更可以让小型团队迅速地进行技术

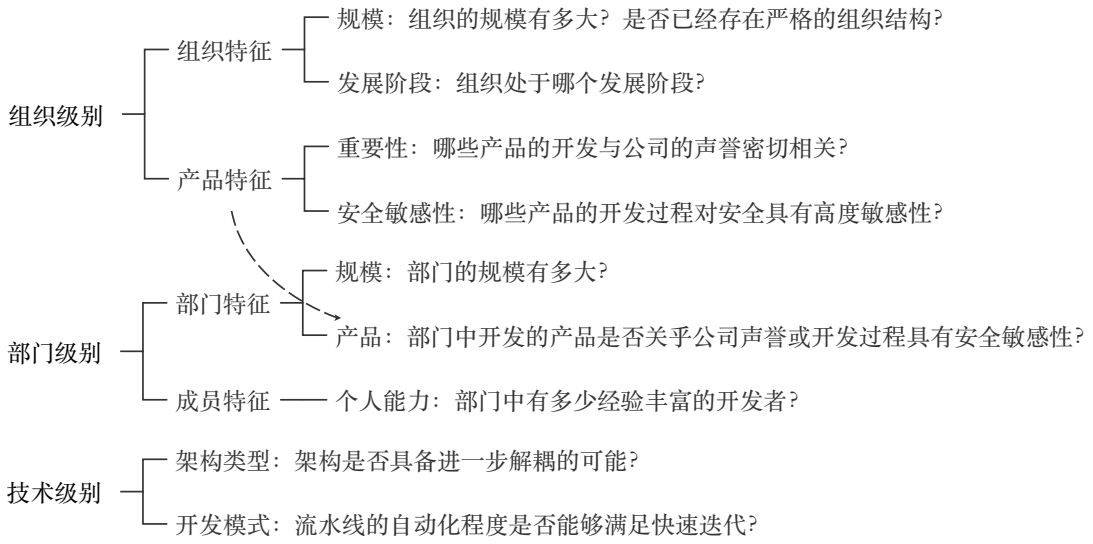


图 6-1: 改进的小型团队决策框架 *ImDMFST*

积累并且快速地进行产品交付。规模较大的 *C*1 与 *C*2 只在一些特定部门或者特定的开发流程中采用小型团队，他们认为小型团队不可以被盲目地在整个组织内部进行推广，那可能会造成组织管理的一些混乱。组织级别的另一个维度是**产品特征**，产品的重要性和安全敏感性对于小型团队的使用存在影响。*C*2 在开发一款数据库核心产品时，由于这款产品作为 *C*2 的主打产品之一且其客户包含政府机构，因此小型团队并没有被用于这款产品的开发。

部门级别

这个级别包含了**部门特征**和**成员特征**两个考虑因素。首先，组织级别的产品特征在部门级别被细化到每个部门开发的产品是否会影响公司的声誉与是否具有安全敏感性。例如，*C*1 中某个安全部门为公司其他部门开发安全工具时，通常会使用一个大型团队进行整体框架的开发，但是对于一些细节，由与不同部门对接的小型团队自己完成。这一层级还包含了**成员特征**这个新的维度。**成员特征**在表4-10中体现为“**需要确保每个小型团队中有足够的（超过 *50%*）经验丰富的开发者**”），结合三家公司的人才管理，这一点考虑因素被重新纳入部门级别。经验丰富的开发者组成的小型团队更具有创造力，*C*3 中的存在一个高级开发者小团队，他们会针对产品中的技术难题进行公关，并定期向其他团队分享最新技术。

技术级别

除了 *DMFST* 提及的“软件架构是否支持进一步解耦”外，开发模式也是一个重要的维度。*C*2 中的部分产品仍然使用瀑布式的软件开发模式，迭代的周期大约在三到六个月，虽然开发流程中使用了一些自动化的流水线，但是小型团队在这个开发模式下很难发挥它们的作用。

6.2 对改进的小型团队决策框架的专家评估结果

层次分析法的专家调查被用于对 *ImDMFST* 进行评价打分（如表6-1）。

表 6-1: 专家打分列表

	E1	E2	E3	E4	E5	E6	E7
组织特征：产品特征	3	1	$\frac{1}{3}$	7	3	$\frac{1}{3}$	9
组织特征：部门特征	3	7	1	$\frac{1}{3}$	7	1	3
组织特征：成员特征	7	7	$\frac{1}{7}$	1	3	$\frac{1}{3}$	9
组织特征：架构类型	3	3	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{5}$	3
组织特征：开发模式	3	9	1	3	5	$\frac{1}{5}$	6
产品特征：部门特征	3	7	3	$\frac{1}{3}$	3	3	$\frac{1}{6}$
产品特征：成员特征	5	9	$\frac{1}{7}$	$\frac{1}{3}$	3	1	3
产品特征：架构类型	1	3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$
产品特征：开发模式	3	9	3	3	3	$\frac{1}{3}$	$\frac{1}{3}$
部门特征：成员特征	5	1	$\frac{1}{7}$	3	$\frac{1}{3}$	$\frac{1}{3}$	3
部门特征：架构类型	3	$\frac{1}{3}$	$\frac{1}{5}$	3	$\frac{1}{7}$	$\frac{1}{5}$	1
部门特征：开发模式	1	1	$\frac{1}{3}$	5	1	$\frac{1}{5}$	$\frac{1}{3}$
成员特征：架构类型	$\frac{1}{3}$	$\frac{1}{5}$	7	3	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{3}$
成员特征：开发模式	$\frac{1}{3}$	1	7	5	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
架构类型：开发模式	1	5	3	3	7	1	1

层次结构的生成

ImDMFST 的目标在于指导企业在软件开发中进行小型团队实践。第6.1节详细描述了 *ImDMFST*，基于图6-1，本文生成了层次分析法中使用的层次结构（图6-2）。

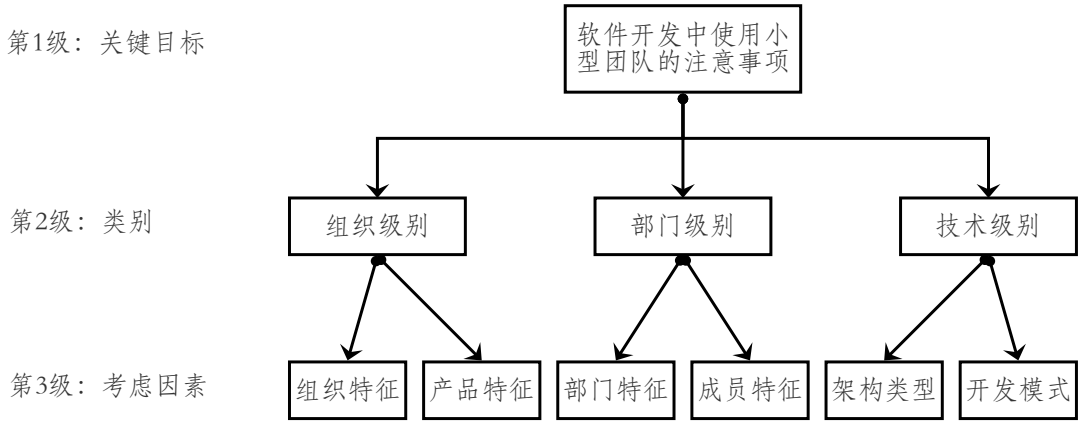


图 6-2: 由 *ImDMFST* 形成的层次结构

判断矩阵的构造与因素权重的计算

从七位专家对六个考虑因素的两两打分，构造了七个判断矩阵 P_i，并分别计算了每个专家的因素权重矩阵 W_i。以专家 E2 为例：

$$P_2 = \begin{bmatrix} 1 & 1 & 7 & 7 & 3 & 9 \\ 1 & 1 & 7 & 9 & 3 & 9 \\ \frac{1}{7} & \frac{1}{7} & 1 & 1 & \frac{1}{3} & 1 \\ \frac{1}{7} & \frac{1}{9} & 1 & 1 & \frac{1}{5} & 1 \\ \frac{1}{3} & \frac{1}{3} & 3 & 5 & 1 & 5 \\ \frac{1}{9} & \frac{1}{9} & 1 & 1 & \frac{1}{5} & 1 \end{bmatrix}, \quad W_2 = \begin{bmatrix} 0.3521 \\ 0.3660 \\ 0.0464 \\ 0.0416 \\ 0.1543 \\ 0.0396 \end{bmatrix} \tag{6-1}$$

专家打分一致性检验

选取 $RI = 1.26$（表 3-7）进行一致性检验，各个专家打分的随机一致性比率 CR_i 如表 6-2所示。根据等式 $CR = \frac{\sum_{i=1}^{7} CR_i}{7}$ 计算出七位专家群体决策的随机一致性比率 $CR = 0.0625$，这说明专家对考虑因素的打分具有满意的一致性。

表 6-2: 专家打分的最大的特征值与随机一致性比率

	专家 E1	专家 E2	专家 E3	专家 E4	专家 E5	专家 E6	专家 E7
$\lambda_{\max}$	6.4314	6.0655	6.5105	6.5834	6.5347	6.0774	6.5553
CR	0.0685	0.0104	0.0810	0.0926	0.0849	0.0123	0.0881

层次总排序

根据七位专家的因素权重（如表6-3）和他们$CMMI$评估经验Ex（表3-6）计算得到的综合权重如表6-4所示。

表 6-3: 七位专家评分的因素权重

	E1	E2	E3	E4	E5	E6	E7
组织特征	0.3669	0.3521	0.0507	0.1876	0.2390	0.0531	0.4614
产品特征	0.2156	0.3660	0.1193	0.0761	0.1225	0.1302	0.0570
部门特征	0.1576	0.0464	0.0434	0.3564	0.0427	0.0531	0.1453
成员特征	0.0381	0.0416	0.5224	0.2043	0.0608	0.1302	0.0401
架构特征	0.1139	0.1543	0.1945	0.1302	0.4674	0.3167	0.1353
开发模式	0.1080	0.0396	0.0698	0.0454	0.0676	0.3167	0.1610

表 6-4: $ImDMFST$ 中考虑因素的权重

考虑因素	组织级别		部门级别		技术级别	
	组织特征	产品特征	部门特征	成员特征	架构类型	开发模式
权重	0.2774	0.1796	0.0869	0.1325	0.2335	0.0901
	0.4570		0.2194		0.3236	

组织级别的考虑因素在专家打分中的权重值（0.4570）远高于另外两个级别的考虑因素，这表明在专家看来，在软件开发中是否使用小型团队应当首先考虑小型团队与软件组织之间的相互影响。以组织规模和发展阶段为代表的组织特征应该在$ImDMFST$中占据最重要的地位。

技术级别的因素在专家打分中的权重值排在第二位，其中架构类型更是仅次于组织特征的考虑因素。小型团队因为体量小，对开发任务之间的耦合性有较高的敏感度，对架构进行进一步的解耦可以减少团队之间的沟通成本。

部门级别的因素在专家打分的权重值最低，部门需要遵循组织的结构与规则，而具体影响软件开发底层的又属于技术级别的因素。

6.3 本章小结

本章报告了对 *DMFST* 的改进与评估。基于对三家公司软件团队的观察，对 *DMFST* 进行补充与调整，形成了包含组织、部门、技术三个级别六个考虑因素的 *ImDMFST*。基于层次分析法的专家调查被用于对 *ImDMFST* 进行评估，发现组织级别的组织特征在 *ImDMFST* 中的权重值最高，而开发模式在 *ImDMFST* 中的权重值最低。

第七章　总结与展望

7.1　总结

DevOps 与微服务已经被众多企业应用与实施以实现高速、高质量地交付用户价值。DevOps 打破了部门墙，微服务拆分了单体系统，这些都深刻的影响了软件团队在 DevOps 与微服务环境下的组织形式与工作模式。一些企业在应用 DevOps 与微服务的过程中，对使用小型团队进行软件开发进行了广泛的探索。然而，小型团队在 DevOps 与微服务的环境下如何组织与实践、软件团队的日常工作中 DevOps 与微服务发挥了什么样的作用以及小型团队在什么情况下应当被应用到软件开发中这三个问题仍然需要被探究。为了回答这三个问题，本文提出了一种研究软件团队的混合方法，设计并实施了案例研究、民族志研究和专家调查这三个阶段的研究。本文的主要贡献如下：

贡献一 基于软件工程研究方法论相关的重要文献，回顾了近五年（2016-2020）发表在经验软件工程领域顶级期刊与会议上有关软件团队研究的论文，总结了一种研究软件团队的迭代式混合方法 $IMMMST$，设计并实施了一个包含三个阶段的混合研究过程。

贡献二 探索了一种名为微战队的小型团队实践，提供了一个使用小型团队进行软件开发的范例，识别了实践中微战队的三种类别和四个关键活动，分析了微战队带来的收益以及影响收益的一些原因，发现了微战队在实践中仍然存在的三个问题并提出了三点持续改进的建议。

贡献三 从沉浸式的视角探究了 DevOps 与微服务在软件团队中发挥的作用，提出 DevOps 在软件团队中的不完整实施和微服务在软件团队中的滥用是目前企业在 DevOps 与微服务实践中存在的两点主要问题，从团队组织结构方面分析了这些问题出现的原因与影响，讨论了 DevOps 与微服务在团队日常工作中的弱关联性、DevOps 相关概念的争论以及在工业界进行实践创新面临的两个挑战。

贡献四 基于对微战队的案例研究提出了一种软件开发中的小型团队决策框架 $DMFST$，通过对三家公司的参与观察对其进行了改进并使用专家调查对改

进的小型团队决策框架 *ImDMFST* 进行了评估与打分。

7.2 展望

本文基于提出的研究软件团队的迭代式混合方法，对 DevOps 与微服务环境下的软件团队进行了探索，重点关注了软件开发中的小型团队。本文的研究分为三个阶段，每一阶段的研究都会为之后的研究阶段提供反馈。三个阶段研究的过程与结果可以继续为之后的研究提供反馈，带来了两点比较研究的机遇，具体如下：

机遇一 *ImDMFST* 综合了软件团队和专家的意见，给企业在软件开发中应用与实施小型团队实践提供了参考，但在不同环境下应用与实施这个框架需要通过大规模的实地研究进行比较。

机遇二 民族志研究发现了 DevOps 与微服务在软件团队中应用与实施的一些问题，这些问题是在三家公司中被发现的，对这三家公司的软件开发过程数据、产品数据等进行比较分析能够使研究更加深入。

除了比较研究的机遇之外，本文提及的 DevOps 成熟度、面向微服务的分解方法、小型团队中的成员关系等话题也值得在未来的研究中进一步的讨论。

致　　谢

从 2014 年到 2021 年，我在南京大学度过了人生中正值青春的七年，当落笔至此，四年的本科和三年的研究生生涯就已经进入了尾声。这七年中，我曾经懵懂无知，也曾经有过彷徨，曾经失落黯然神伤，也曾经喜悦不可自拔，曾经见过凌晨三四五六时的实验室外的天空，也曾经晚上七八九十点在操场上挥洒汗水。七年的时光，在我的生命中刻下了深深的印记，也让我逐渐走向成熟，成为一个可以为社会带来贡献的人。在这个过程中，老师、家人、朋友与同学们的帮助与支持起到了至关重要的作用。在这里，在我即将离开校园之际，我要诚挚地向你们表示感谢。

首先要感谢我的导师张贺教授，本科三年级时与张贺教授的交流仍然历历在目，转瞬已经过去四年，这四年间，是张教授带领我走进了学术科研的大门，他严谨的治学、高超的学术造诣以及积极的科研态度无不给我留下了深刻地印象并为我以后的学习与工作提供了一个卓越的模版。在张教授的指导下，我也在科研上取得了一些成绩，无论是中文期刊还是国际会议，每一篇论文的背后，都有着张教授对字句的斟酌，每一次论文的发表以及每一次在国际会议上的论文汇报，都让我的眼界更加开阔。此外我还要感谢软件研发效能实验室的其他三位老师。邵栋副教授作为国内敏捷领域的专家，在我的研究中给予了很多帮助，加深了我对敏捷方法、软件团队以及软件工程教育的理解；荣国平副研究员从我研究生刚入学就指导我在 DevOps、软件过程改进方面的研究；匡宏宇助理研究员在我毕业论文开题的过程中提供了很多让我受益匪浅的建议，并且在论文写作指导上分享了很多经验。科研与论文之外，我还要感谢南京大学软件学院的其他老师们。丰富的课程如潘敏学老师的软件设计、冯桂焕老师的人机交互、李传艺老师的高级算法等极大地拓宽了我的眼界；七年中黄蕾、张田田、张蕾几位辅导员老师在校园生活也对我给予了极大的帮助。澳大利亚阿德莱德大学的 Muhammad Ali Babar 教授、挪威科技大学的李京悦教授和 Letizia Jaccheri 教授、澳大利亚凯斯林大学的沈海峰教授、智利康塞普西翁大学的李铮老师、华为的陈连平老师、星环科技的刘汪根老师、思目创意的杜波师兄在我三年的研究与学习生涯中都为我提供了很多宝贵的建议并分享了很多

实用的经验，我从中获益良多。

然后我要感谢我的朋友们与同学们。黄鑫与周鑫两位师兄在科研和生活中对我帮助良多，我发表的论文几乎都是与两位师兄合作完成，两位师兄对软件工程研究方法的深刻见解让我在研究的过程中轻松许多。刘博涵师兄和毛润丰博士一直与我一同探索 DevOps 和相关的话题，这是我毕业论文关注的重点之一。李杉杉师姐则在我论文的另一关注点微服务上给了我很多启发与建议。在这里也要感谢张瑾师姐和 Nora Tomas 在我短期的挪威访学期间对我的合作和科研工作的支持与帮助。我也要感谢在之前的研究、实习与毕业论文的写作过程中，对我提供过帮助的华为、星环科技、思目创意的师兄师姐们，这些经历会成为以后我在学习和工作的过程中值得回忆的部分。在南京大学学习生活的这些年，也离不开室友们的鼓励与支持，黄迪璇和胡文两位已经毕业的室友曾在课程学习和日常生活中帮助我很多，良好的寝室氛围也让我回忆颇多，还有，如果你们看到了，记得请客吃饭。和我一同毕业的室友戴启铭，大四进入实验室时，没有想到半年后会和他成为室友，三年里我们在研究中相互扶持与进步，也合作完成了一些论文的工作，在生活中他对我也帮助很多。与杨岚心一起在球场挥洒汗水、与钟陈星一起在校园里丈量时间的长度，在我最后一年的研究生生活中都是对心灵的放松，为学习与工作提供了充足的动力。与史坚、王康三人虽然分别在南京北京，三人中有即将毕业、有工作已经快一年、也有还在努力攻读博士学位，但我们从中学开始的友谊仍然新鲜，相互的鼓励一直是我前进的动力。

最后我要感谢我的父母，是他们在生活中支持和鼓励着我，让我能够心无旁骛地完成七年的本科和研究生求学生涯。无论以后在哪学习工作，家永远是那永恒的港湾。

三年与七年，时光如白驹过隙，但这些年里发生的故事在以后很长的时间中仍会记忆犹新。十九载学生生涯即将在今年告一段落，但 “活到老学到老”，南京大学的求学经历将使我受益终生。

参考文献

[1] 黄璜, 张贺, 邵栋. 自动化工具对中国 DevOps 实践的影响 [J]. 软件学报, 2019, 30(10) : 3056 – 3070.

[2] SNYDER A. Encapsulation and inheritance in object-oriented programming languages[C] // Proceedings of the 1st International Conference on Object-Oriented Programming Systems, Languages and Applications. 1986 : 38 – 45.

[3] PAPAZOGLOU M P. Service-oriented computing: Concepts, characteristics and directions[C] // Proceedings of the 4th International Conference on Web Information Systems Engineering. 2003 : 3 – 12.

[4] BECK K, GRENNING J, MARTIN R C, et al. Manifesto for Agile software development[EB/OL]. Agile Alliance, [2021/3/1]. http://agilemanifesto.org/iso/en/manifesto.html.

[5] ALLSPAW J, HAMMOND P. 10+ deploys per day: Dev and Ops cooperation at Flickr[C] // Velocity: Web Performance and Operations Conference. 2009.

[6] FOWLER M, LEWIS J. Microservices: A definition of this new architectural term[EB/OL]. 2014 [2021/3/1]. https://martinfowler.com/articles/microservices.html.

[7] GOOGLE. 2010 年以来 DevOps 与微服务的 Google 搜索趋势（按主题）[EB/OL]. Google, [2021/3/1]. https://trends.google.com/trends/explore?date=2010-01-01%202021-02-01&q=%2Fm%2F0c3tq11,%2Fm%2F011spz0k.

[8] ARTAC M, BOROVSSAK T, DI NITTO E, et al. DevOps: Introducing infrastructure-as-code[C] // Proceedings of the 39th International Conference on Software Engineering: Companion. 2017 : 497 – 498.

[9] DYCK A, PENNERS R, LICHTER H. Towards definitions for release engineering and DevOps[C] // Proceedings of the 3rd International Workshop on Release Engineering. 2015 : 3 – 3.

[10] BROWN A, STAHNKE M, KERSTEN N. State of DevOps 2020[R]. London : Puppet, 2020.

[11] 张贺, 刘博涵, 荣国平, et al. 2019 DevOps · 中国调查报告 [R]. 南京 : 南京大学, 2019.

[12] TAIBI D, LENARDUZZI V, PAHL C. Continuous architecting with microservices and DevOps: A systematic mapping study[C] // Proceedings of the 8th International Conference on Cloud Computing and Services Science. 2018 : 126 – 151.

[13] CHEN L. Microservices: Architecting for continuous delivery and DevOps[C] // Proceedings of the 15th International Conference on Software Architecture. 2018 : 39 – 46.

[14] RICHARDS M. Microservices vs. Service-Oriented Architecture[R]. Online : O'Reilly, 2016.

[15] BALALAIE A, HEYDARNOORI A, JAMSHIDI P. Microservices architecture enables DevOps: Migration to a cloud-native architecture[J]. IEEE Software, 2016, 33(3) : 42 – 52.

[16] ZIMMERMANN O. Microservices tenets[J]. Computer Science-Research and Development, 2017, 32(3) : 301 – 310.

[17] FORD N. The state of microservices maturity[R]. Beijing, Boston, Farnham, Sebastopol, Tokyo : O'Reilly, 2020.

[18] RICHARDSON D, SINGH D, EXNER K. Using DevOps, microservices, & serverless to accelerate innovation[R]. Online : Amazon Web Services, 2018.

[19] AZURE M. DevTest and DevOps for microservice solutions[EB/OL]. Microsoft, [2021/3/1].

https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/dev-test-microservice.

[20] CLOUD G. DevOps tech: Architecture[EB/OL]. Google, [2021/3/1]. https://cloud.google.com/solutions/devops/devops-tech-architecture.

[21] LWAKATARE L E, KILAMO T, KARVONEN T, et al. DevOps in practice: A multiple case study of five companies[J]. Information and Software Technology, 2019, 114 : 217 – 230.

[22] DÍAZ J, PEREZ J E, YAGUE A, et al. Devops in practice–A preliminary analysis of two multinational companies[C] // Proceedings of the 20th International Conference on Product-Focused Software Process Improvement. 2019 : 323 – 330.

[23] ZHANG H, LI S, JIA Z, et al. Microservice architecture in reality: An industrial inquiry[C] // Proceedings of the 16th International Conference on Software Architecture. 2019 : 51 – 60.

[24] FRITZSCH J, BOGNER J, WAGNER S, et al. Microservices migration in industry: Intentions, strategies, and challenges[C] // Proceedings of the 35th International Conference on Software Maintenance and Evolution. 2019 : 481 – 490.

[25] MCCARTHY J, MCCARTHY M. Software for your head: Core protocols for creating and maintaining shared vision[M]. Boston : Addison-Wesley, 2002.

[26] 崔海涛, 章程, 丁翔, et al. 面向微服务架构的开发组织适应性评估框架研究[J]. 软件学报, 2021, 32(5) : 1256 – 1283.

[27] HOEGL M. Smaller teams–better teamwork: How to keep project teams small[J]. Business Horizons, 2005, 48(3) : 209 – 214.

[28] WU L, WANG D, EVANS J A. Large teams develop and small teams disrupt science and technology[J]. Nature, 2019, 566(7744) : 378 – 382.

[29] STOREY M-A, ERNST N A, WILLIAMS C, et al. The who, what, how of software engineering research: A socio-technical framework[J]. Empirical Software Engineering, 2020, 25(5) : 4097 – 4129.

[30] FRONZA I, WANG X. Towards an approach to prevent social loafing in software development teams[C] // Proceedings of the 11th International Symposium on Empirical Software Engineering and Measurement. 2017 : 241 – 246.

[31] PAASIVAARA M, BEHM B, LASSENIUS C, et al. Large-scale agile transformation at Ericsson: A case study[J]. Empirical Software Engineering, 2018, 23(5) : 2550 – 2596.

[32] YATES R, POWER N, BUCKLEY J. Characterizing the transfer of program comprehension in onboarding: An information-push perspective[J]. Empirical Software Engineering, 2020, 25(1) : 940 – 995.

[33] TASHAKKORI A, TEDDLIE C, TEDDLIE C B. Mixed methodology: Combining qualitative and quantitative approaches[M]. California : Sage Publications, 1998.

[34] STOL K-J, FITZGERALD B. The ABC of software engineering research[J]. ACM Transactions on Software Engineering and Methodology, 2018, 27(3) : 1 – 51.

[35] LETHBRIDGE T C, SIM S E, SINGER J. Studying software engineers: Data collection techniques for software field studies[J]. Empirical Software Engineering, 2005, 10(3) : 311 – 341.

[36] GOTEL O, LEIP D. Agile software development meets corporate deployment procedures: Stretching the agile envelope[C] // Proceedings of the 8th International Conference on Extreme Programming and Agile Processes in Software Engineering. 2007 : 24 – 27.

[37] LEAU Y B, LOO W K, THAM W Y, et al. Software development life cycle: Agile vs traditional approaches[C] // Proceedings of the 2nd International Conference on Information and Network Technology. 2012 : 162 – 167.

[38] DE FRANÇA B B N, JERONIMO H, TRAVASSOS G H. Characterizing DevOps by hearing multiple voices[C] // Proceedings of the 30th Brazilian Symposium on Software Engineering. 2016 : 53 – 62.

[39] 荣国平, 张贺, 邵栋, et al. DevOps: 原理、方法与实践 [M]. 北京 : 机械工业出版社, 2017.

[40] 戴启铭, 毛润丰, 黄璜, et al. DevSecOps:DevOps 下实现持续安全的实践探索 [J/OL]. 软件学报, 2021, 0(0) : 1 – 22. http://dx.doi.org/10.13328/j.cnki.jos.006276.

[41] BASS L, WEBER I, ZHU L. DevOps: A software architect's perspective[M]. Boston : Addison-Wesley Professional, 2015.

[42] HUMBLE J, FARLEY D. Continuous delivery: Reliable software releases through build, test, and deployment automation[M]. London : Pearson Education, 2010.

[43] MYRBAKKEN H, COLOMO-PALACIOS R. DevSecOps: A multivocal literature review[C] // Proceedings of the 17th International Conference on Software Process Improvement and Capability Determination. 2017 : 17 – 29.

[44] MAO R, ZHANG H, DAI Q, et al. Preliminary findings about DevSecOps from grey literature[C] // Proceedings of the 20th International Conference on Software Quality, Reliability and Security. 2020 : 450 – 457.

[45] TOMAS N, LI J, HUANG H. An empirical study on culture, automation, measurement, and sharing of DevSecOps[C] // Proceedings of the 2019 International Conference on Cyber Security and Protection of Digital Services. 2019 : 404 – 411.

[46] RONG G, JIN Z, ZHANG H, et al. DevDocOps: Enabling continuous documentation in alignment with DevOps[J]. Software: Practice and Experience, 2020, 50(3) : 210 – 226.

[47] LUZ W P, PINTO G, BONIFÁCIO R. Adopting DevOps in the real world: A theory, a model, and a case study[J]. Journal of Systems and Software, 2019, 157 : 110384.

[48] JOSEPH C T, CHANDRASEKARAN K. Straddling the crevasse: A review of microservice software architecture foundations and recent advancements[J]. Software: Practice and Experience, 2019, 49(10) : 1448 – 1484.

[49] KECSKEMETI G, MAROSI A C, KERTESZ A. The ENTICE approach to decompose monolithic services into microservices[C] // Proceedings of the 14th International Conference on High Performance Computing and Simulation. 2016 : 591 – 596.

[50] LI S, ZHANG H, JIA Z, et al. A dataflow-driven approach to identifying microservices from monolithic applications[J]. Journal of Systems and Software, 2019, 157 : 110380.

[51] DI FRANCESCO P, LAGO P, MALAVOLTA I. Architecting with microservices: A systematic mapping study[J]. Journal of Systems and Software, 2019, 150 : 77 – 97.

[52] JAMSHIDI P, PAHL C, MENDONÇA N C, et al. Microservices: The journey so far and challenges ahead[J]. IEEE Software, 2018, 35(3) : 24 – 35.

[53] SOLDANI J, TAMBURRI D A, VAN DEN HEUVEL W-J. The pains and gains of microservices: A Systematic grey literature review[J]. Journal of Systems and Software, 2018, 146 : 215 – 232.

[54] THÖNES J. Microservices[J]. IEEE software, 2015, 32(1) : 116 – 116.

[55] EVANS E. Domain-driven design: Tackling complexity in the heart of software[M]. Boston : Addison-Wesley Professional, 2004.

[56] 钟陈星, 李杉杉, 张贺, et al. 限界上下文视角下的微服务粒度评估 [J]. 软件学报, 2019, 30(10) : 3227 – 3241.

[57] GYSEL M, KÖLBENER L, GIERSCHE W, et al. Service cutter: A systematic approach to service decomposition[C] // Proceedings of the 5th European Conference on Service-Oriented and Cloud Computing. 2016 : 185 – 200.

[58] BOZAN K, LYYTINEN K, ROSE G M. How to transition incrementally to microservice architecture[J]. Communications of the ACM, 2020, 64(1) : 79 – 85.

[59] WASEEM M, LIANG P, SHAHIN M. A systematic mapping study on microservices architecture in DevOps[J]. Journal of Systems and Software, 2020, 170 : 110798.

[60] THANH T Q, COVACI S, MAGEDANZ T, et al. Embedding security and privacy into the development and operation of cloud applications and services[C] // Proceedings of the 17th International Telecommunications Network Strategy and Planning Symposium. 2016 : 31 – 36.

[61] PAHL C, JAMSHIDI P, ZIMMERMANN O. Architectural principles for cloud software[J]. ACM Transactions on Internet Technology, 2018, 18(2) : 1 – 23.

[62] MIGLIERINA M, TAMBURRI D A. Towards omnia: A monitoring factory for quality-aware devops[C] // Proceedings of the 8th International Conference on Performance Engineering Companion. 2017 : 145 – 150.

[63] O'CONNOR R V, ELGER P, CLARKE P M. Continuous software engineering —-A microservices architecture perspective[J]. Journal of Software: Evolution and Process, 2017, 29(11) : e1866.

[64] BALALAIE A, HEYDARNOORI A, JAMSHIDI P. Migrating to cloud-native architectures using microservices: An experience report[C] // Proceedings of the 4th European Conference on Service-Oriented and Cloud Computing. 2015 : 201 – 215.

[65] SHAHIN M, BABAR M A. On the role of software architecture in DevOps transformation: An industrial case study[C] // Proceedings of the 14th International Conference on Software and System Processes. 2020 : 175 – 184.

[66] CHOUDHRY A, PREMCHAND A. Microservices and DevOps for optimal benefits from IoT in manufacturing[C] // Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications. 2020 : 375 – 384.

[67] RADEMACHER F, SACHWEH S, ZÜNDORF A. Deriving microservice code from underspecified domain models using DevOps-enabled modeling languages and model transformations[C] // Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications. 2020 : 229 – 236.

[68] SHARP H, DITTRICH Y, DE SOUZA C R. The role of ethnographic studies in empirical software engineering[J]. IEEE Transactions on Software Engineering, 2016, 42(8) : 786 – 804.

[69] ZHANG H, HUANG H, SHAO D, et al. Fireteam: A small-team development practice in industry[C] // Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020 : 1365 – 1375.

[70] HIGHSMITH J, COCKBURN A. Agile software development: The business of innovation[J]. Computer, 2001, 34(9) : 120 – 127.

[71] ZANONI J C, RAMOS M P, TACLA C A, et al. A semi-automatic source code documentation method for small software development teams[C] // Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design. 2011 : 113 – 119.

[72] COCKBURN A, HIGHSMITH J. Agile software development, the people factor[J]. Computer, 2001, 34(11) : 131 – 133.

[73] NERUR S, MAHAPATRA R, MANGALARAJ G. Challenges of migrating to agile methodologies[J]. Communications of the ACM, 2005, 48(5) : 72 – 78.

[74] LINDSJØRN Y, SJØBERG D I, DINGSØYR T, et al. Teamwork quality and project success in software development: A survey of agile development teams[J]. Journal of Systems and Software, 2016, 122 : 274 – 286.

[75] LINDSJØRN Y, BERGERSEN G R, DINGSØYR T, et al. Teamwork quality and team performance: Exploring differences between small and large agile projects[C] // Proceedings of the International Conference on Agile Software Development. 2018 : 267 – 274.

[76] HILTON M, BEGEL A. A study of the organizational dynamics of software teams[C] // Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice. 2018 : 191 – 200.

[77] DYBÅ T, DINGSØYR T. Empirical studies of agile software development: A systematic review[J]. Information and Software Technology, 2008, 50(9–10): 833 – 859.

[78] DA SILVA F Q, FRANÇA A C C, SUASSUNA M, et al. Team building criteria in software projects: A mix-method replicated study[J]. Information and Software Technology, 2013, 55(7): 1316 – 1340.

[79] LEE S, YONG H-S. Agile software development framework in a small project environment.[J]. Journal Information Process System, 2013, 9(1): 69 – 88.

[80] SEPTIAN W, GATA W. Software development framework on small team using Agile Framework for Small Projects (AFSP) with neural network estimation[C] // Proceedings of the 11th International Conference on Information and Communication Technology and System. 2017: 259 – 264.

[81] KNIBERG H, IVARSSON A. Scaling agile@ spotify with tribes, squads, chapters & guilds[R]. Stockholm: Spotify, 2012.

[82] SALAMEH A, BASS J. Influential factors of aligning spotify squads in mission-critical and offshore projects–A longitudinal embedded case study[C] // Proceedings of the International Conference on Product-Focused Software Process Improvement. 2018: 199 – 215.

[83] FERNANDES T. Spotify Squad framework – Part I & Part II[R]. Stockholm: Spotify, 2017.

[84] SALAMEH A, BASS J M. Spotify tailoring for promoting effectiveness in cross-functional autonomous squads[C] // Proceedings of the International Conference on Agile Software Development. 2019: 20 – 28.

[85] SIGNORETTI I, MARCZAK S, SALERNO L, et al. Boosting agile by using user-centered design and lean startup: A case study of the adoption of the combined approach in software development[C] // Proceedings of the 13th International Symposium on Empirical Software Engineering and Measurement. 2019: 1 – 6.

[86] NELSON N, BRINDESCU C, MCKEE S, et al. The life-cycle of merge conflicts: Processes, barriers, and strategies[J]. Empirical Software Engineering, 2019, 24(5): 2863 – 2906.

[87] SEDANO T, RALPH P, PÉRAIRE C. Sustainable software development through overlapping pair rotation[C] // Proceedings of the 10th International Symposium on Empirical Software Engineering and Measurement. 2016: 1 – 10.

[88] WANG Y, REDMILES D. Cheap talk, cooperation, and trust in global software engineering[J]. Empirical Software Engineering, 2016, 21(6): 2233 – 2267.

[89] CHATZIPETROU P, ŠMITE D, VAN SOLINGEN R. When and who leaves matters: Emerging results from an empirical study of employee turnover[C] // Proceedings of the 12th International Symposium on Empirical Software Engineering and Measurement. 2018: 1 – 4.

[90] YIN R K. Applications of case study research[M]. California: Sage Publications, 2011.

[91] FETTERMAN D M. Ethnography: Step-by-step[M]. California: Sage Publications, 2009.

[92] ZHANG H, HUANG X, ZHOU X, et al. Ethnographic research in software engineering: A critical review and checklist[C] // Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019: 659 – 670.

[93] JOBLIN M, APEL S, MAUERER W. Evolutionary trends of developer coordination: A network approach[J]. Empirical Software Engineering, 2017, 22(4): 2050 – 2094.

[94] BATISTA A C D, DE SOUZA R M, DA SILVA F Q, et al. Teamwork quality and team success in software development: A non-exact replication study[C] // Proceedings of the 14th International Symposium on Empirical Software Engineering and Measurement. 2020: 1 – 11.

[95] SAKHNINI V, MICH L, BERRY D M. Group versus individual use of power-only EPMcreate as a creativity enhancement technique for requirements elicitation[J]. Empirical Software Engineering, 2017, 22(4): 2001 – 2049.

[96] MASOOD Z, HODA R, BLINCOE K. How agile teams make self-assignment work: A grounded theory study[J]. Empirical Software Engineering, 2020, 25(6): 4962 – 5005.

[97] SJØBERG D I. An empirical study of WIP in kanban teams[C] // Proceedings of the 12th International Symposium on Empirical Software Engineering and Measurement. 2018: 1 – 8.

[98] BRITTO R, ŠMITE D, DAMM L-O. Experiences from measuring learning and performance in large-scale distributed software development[C] // Proceedings of the 10th International Symposium on Empirical Software Engineering and Measurement. 2016: 1 – 6.

[99] FU C, ZHANG H, HUANG X, et al. A review of meta-ethnographies in software engineering[C] // Proceedings of the 23th International Conference on Evaluation and Assessment in Software Engineering. 2019: 68 – 77.

[100] STOL K-J, RALPH P, FITZGERALD B. Grounded theory in software engineering research: A critical review and guidelines[C] // Proceedings of the 38th International Conference on Software Engineering. 2016: 120 – 131.

[101] HUANG X, ZHANG H, ZHOU X, et al. Synthesizing qualitative research in software engineering: A critical review[C] // Proceedings of the 40th International Conference on Software Engineering. 2018: 1207 – 1218.

[102] CRUZES D S, DYBA T. Recommended steps for thematic synthesis in software engineering[C] // Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement. 2011: 275 – 284.

[103] RUNESON P, HÖST M. Guidelines for conducting and reporting case study research in software engineering[J]. Empirical Software Engineering, 2009, 14(2): 131 – 164.

[104] SHARP H, DESOUZA C, DITTRICH Y. Using ethnographic methods in software engineering research[C] // Proceedings of the 32nd International Conference on Software Engineering. 2010 : 491 – 492.

[105] YIGIT T, ISIK A H, INCE M. Web-based learning object selection software using analytical hierarchy process[J]. IET Software, 2014, 8(4) : 174 – 183.

[106] KHAN A A, SHAMEEM M. Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process[J]. Journal of Software: Evolution and Process, 2020, 32(10) : e2263.

[107] RUNESON P, HOST M, RAINER A, et al. Case study research in software engineering: Guidelines and examples[M]. New Jersey : John Wiley & Sons, 2012.

[108] BASKARADA S. Qualitative case study guidelines[J]. Social Science Electronic Publishing, 2015, 19(40) : 1 – 25.

[109] BOWEN G A. Document analysis as a qualitative research method[J]. Qualitative Research Journal, 2009.

[110] CURASI C F. A critical exploration of face-to-face interviewing vs. computer-mediated interviewing[J]. International Journal of Market Research, 2001, 43(4) : 361 – 375.

[111] HOVE S E, ANDA B. Experiences from conducting semi-structured interviews in empirical software engineering research[C] // Proceedings of the 11th IEEE International Software Metrics Symposium. 2005 : 10 – 23.

[112] FOWLER F J. Applied social research methods : Survey research methods[M]. California : Sage Publications, 2013.

[113] CIOLKOWSKI M, LAITENBERGER O, VEGAS S, et al. Practical experiences in the design and conduct of surveys in empirical software engineering[G] // Empirical Methods and Studies in Software Engineering. 2003 : 104 – 128.

[114] PETTICREW M, ARAI L, ROBERTS H, et al. Testing methodological guidance on the conduct of narrative synthesis in systematic reviews[J]. Evaluation, 2009, 15(1): 49 – 73.

[115] HERBERT S. For ethnography[J]. Progress in Human Geography, 2000, 24(4): 550 – 568.

[116] GERARD FORSEY M. Ethnography as participant listening[J]. Ethnography, 2010, 11(4): 558 – 572.

[117] ROMANO S, FUCCI D, SCANNIELLO G, et al. Findings from a multi-method study on test-driven development[J]. Information and Software Technology, 2017, 89: 64 – 77.

[118] SHAH H, NERSESSIAN N J. Cultural models and their interplay in global software engineering practice[C] // Proceedings of the 10th International Conference on Global Software Engineering. 2015: 13 – 22.

[119] BJØRN P, ESBENSEN M, JENSEN R E, et al. Does distance still matter? Revisiting the CSCW fundamentals on distributed collaboration[J]. ACM Transactions on Computer-Human Interaction, 2014, 21(5): 27.

[120] KARN J S, COWLING A J. A study of the effect of disruptions on the performance of software engineering teams[C] // Proceedings of the 4th International Symposium on Empirical Software Engineering. 2005: 417 – 425.

[121] HAMMERSLEY M, ATKINSON P. Ethnography: Principles in practice[M]. Oxford: Routledge, 2007.

[122] SALVADOR T, BELL G, ANDERSON K. Design ethnography[J]. Design Management Journal (Former Series), 1999, 10(4): 35 – 41.

[123] STRAUSS A, CORBIN J. Basics of qualitative research techniques[M]. Pennsylvania: Citeseer, 1998.

[124] AL-HARBI K M A-S. Application of the AHP in project management[J]. International journal of project management, 2001, 19(1): 19 – 27.

[125] BRANNICK M T, PRINCE C. An overview of team performance measurement[G] // Team Performance Assessment and Measurement. Oxford : Taylor & Francis, 1997 : 15 – 28.

[126] PETERSEN K. Measuring and predicting software productivity: A systematic map and review[J]. Information and Software Technology, 2011, 53(4) : 317 – 343.

[127] OLIVEIRA E, VIANA D, CRISTO M, et al. How have software engineering researchers been measuring software productivity? A systematic mapping study[C] // Proceedings of the 19th International Conference on Enterprise Information Systems : Vol 2. 2017 : 76 – 87.

[128] FAGERHOLM F, IKONEN M, KETTUNEN P, et al. How do software developers experience team performance in lean and agile environments?[C] // Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. 2014 : 1 – 10.

[129] HERBSLEB J D, GRINTER R E. Architectures, coordination, and distance: Conway's law and beyond[J]. IEEE Software, 1999, 16(5) : 63 – 70.

[130] CONWAY M E. How do committees invent[J]. Datamation, 1968, 14(4) : 28 – 31.

[131] HERBSLEB J D, GRINTER R E. Splitting the organization and integrating the code: Conway's law revisited[C] // Proceedings of the 21st International Conference on Software Engineering. 1999 : 85 – 95.

[132] KIRKHOPE C. But...no one told me to do it[R]. Edinburgh : Lean Agile Scotland, 2017.

[133] MARQUET L D. 你就是艇长 [M]. 广州 : 广东人民出版社, 2014.

[134] HACKMAN J R. Collaborative intelligence: Using teams to solve hard problems[M]. California : Berrett-Koehler Publishers, 2011.

[135] HUNDERMARK P, KALTENECKER S. Why do we need self-organising teams?[EB/OL]. InfoQ, 2014 [2021/3/1]. https://www.infoq.com/articles/why-need-self-organizing-teams.

[136] RADEMACHER F, SORGALLA J, WIZENTY P N, et al. Microservice architecture and model-driven development: Yet singles, soon married (?)[C] // Proceedings of the 19th International Conference on Agile Software Development. 2018 : 1 – 5.

[137] RONG G, JIN Z, ZHANG H, et al. DevDocOps: Towards automated documentation for DevOps[C] // Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice. 2019 : 243 – 252.

[138] 金泽锋, 张佑文, 叶文华, et al. 面向完整价值交付的文档 DevOps 应用研究[J]. 软件学报, 2019, 30(10) : 3127 – 3147.

[139] BERTOLINO A, ANGELIS G D, GUERRIERO A, et al. DevOpRET: Continuous reliability testing in DevOps[J]. Journal of Software: Evolution and Process, 2020 : e2298.

[140] MACDONALD N, HAIGHT C. DevOpsSec: Creating the agile triangle[R]. Online : Gartner, 2012.

[141] CARTER K. Francois raynaud on DevSecOps[J]. IEEE Software, 2017, 34(5) : 93 – 96.

[142] ASHENDEN D, OLLIS G. Putting the sec in DevSecOps: Using social practice theory to improve secure software development[C] // Proceedings of the 11th New Security Paradigms Workshop. 2020 : 34 – 44.

[143] RIGGINS J. DevSecOps: Security automation in enterprise DevOps[EB/OL]. The New Stack, 2018 [2021/3/1]. https://thenewstack.io/devsecops-security-automation-in-enterprise-devops.

[144] VEHENT J. Continuous Security in DevOps[R]. Online : Mozilla, 2018.

[145] SIGLER E. Trends in DevOps: Security[EB/OL]. PagerDuty, 2017 [2021/3/1]. https://www.pagerduty.com/blog/devops-trends-security.

[146] CORNILS B. DevOps and security: Cultural changes to bring Dev, Sec & Ops together[EB/OL]. DZone, 2017 [2021/3/1]. https://dzone.com/articles/devops-and-security-cultural-changes-to-bring-dev.

[147] MCGRAW G. Software security[J]. IEEE Security and Privacy, 2004, 2(2): 80 – 83.

[148] MCGRAW G. Software security: Building security in[M]. Boston: Addison-Wesley Professional, 2006.

[149] RODRÍGUEZ P, MARKKULA J, OIVO M, et al. Survey on agile and lean usage in finnish software industry[C] // Proceedings of the 6th International Symposium on Empirical Software Engineering and Measurement. 2012: 139 – 148.

[150] RIUNGU-KALLIOSAARI L, MÄKINEN S, LWAKATARE L E, et al. DevOps adoption benefits and challenges in practice: A case study[C] // Proceedings of the International Conference on Product-Focused Software Process Improvement. 2016: 590 – 597.

附录 A　文献综述纳入论文列表

表 A-1: 文献综述纳入论文列表

编号	标题	年份	期刊/会议
S1	Teamwork quality and team success in software development: A non-exact replication study	2020	*ESEM*
S2	How agile teams make self-assignment work: A grounded theory study	2020	*EMSE*
S3	On the fulfillment of coordination requirements in open-source software projects: An exploratory study	2020	*EMSE*
S4	What distinguishes great software engineers?	2020	*EMSE*
S5	Code and commit metrics of developer productivity: A study on team leaders perceptions	2020	*EMSE*
S6	A gamification solution for improving Scrum adoption	2020	*EMSE*
S7	Characterizing the transfer of program comprehension in onboarding: An information-push perspective	2020	*EMSE*
S8	Visualizing inter-team coordination	2020	*EASE*
S9	Heterogeneous tailoring approach using the spotify model	2020	*EASE*
S10	Studying onboarding in distributed software teams: A case study and guidelines	2020	*EASE*
S11	Boosting agile by using user-centered design and lean startup: A case study of the adoption of the combined approach in software development	2019	*ESEM*
S12	Effective team onboarding in Agile software development: Techniques and goals	2019	*ESEM*
S13	The life-cycle of merge conflicts: Processes, barriers, and strategies	2019	*EMSE*
S14	The links between agile practices, interpersonal conflict, and perceived productivity	2019	*EASE*
S15	An empirical study of WIP in kanban teams	2018	*ESEM*

续表 A-1

编号	标题	年份	期刊/会议
S16	When and who leaves matters: Emerging results from an empirical study of employee turnover	2018	*ESEM*
S17	Building a collaborative culture: A grounded theory of well succeeded devops adoption in practice	2018	*ESEM*
S18	How does developer interaction relate to software quality? An examination of product development data	2018	*EMSE*
S19	Large-scale agile transformation at Ericsson: A case study	2018	*EMSE*
S20	Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation	2018	*EMSE*
S21	Satisfaction, practices, and influences in agile software development	2018	*EASE*
S22	Towards an approach to prevent social loafing in software development teams	2017	*ESEM*
S23	Team maturity in software engineering teams	2017	*ESEM*
S24	An initial analysis of software engineers'attitudes towards organizational change	2017	*EMSE*
S25	Group versus individual use of power-only EPMcreate as a creativity enhancement technique for requirements elicitation	2017	*EMSE*
S26	Evolutionary trends of developer coordination: A network approach	2017	*EMSE*
S27	Managing the requirements flow from strategy to release in large-scale agile development: A case study at Ericsson	2017	*EMSE*
S28	Recurring opinions or productive improvements–What agile teams actually discuss in retrospectives	2017	*EMSE*
S29	Motivation and autonomy in global software development	2017	*EASE*
S30	How to reduce software development cost with personnel assignment optimization	2017	*EASE*
S31	Adopting continuous delivery and deployment: Impacts on team structures, collaboration and responsibilities	2017	*EASE*
S32	Comparing communication effort within the scrum, scrum with Kanban, XP, and Banana development processes	2017	*EASE*
S33	Virtual team configurations that promote better product quality	2016	*ESEM*

续表 A-1

编号	标题	年份	期刊/会议
S34	Sustainable software development through overlapping pair rotation	2016	*ESEM*
S35	Software project managers' perceptions of productivity factors: Findings from a qualitative study	2016	*ESEM*
S36	Experiences from measuring learning and performance in large-scale distributed software development	2016	*ESEM*
S37	Cheap talk, cooperation, and trust in global software engineering: An evolutionary game theory model with empirical support	2016	*EMSE*
S38	Archetypal personalities of software engineers and their work preferences: A new perspective for empirical studies	2016	*EMSE*
S39	Benefits and limitations of job rotation in software organizations: A systematic literature review	2016	*EASE*
S40	Practice and perception of team code ownership	2016	*EASE*

附录 B　访谈与调研

B.1　第一次微战队访谈问题清单

被访谈成员背景

1. 你在战队中的职责是什么？你在上一个团队中的职责是什么?

被访谈战队背景

2. 你们战队是如何组建的？你们是全功能团队吗，是否是按照独立的特性团队组建的，还是几个微战队组成一个特性团队?

3. （战队中有多少人？）你认为影响战队规模的因素是什么?

4. 战队的负责人是谁？你（们）为什么选择他？（你认为他们呢为什么选择了你？）

5. 战队成立后，你的工作地点是否发生了变化?

6. 架构对战队组建的影响有多大？是软件架构的调整导致了微战队的产生，还是为了适应微战队的模式而调整了软件架构?

7. 如果有人离开战队，你们的应急计划是什么？您认为什么情况会导致这种人员离职?

微战队工作流程

8. 谁负责战队的日常管理，队长或其他人员？你如何看待管理工作对他正常工作的影响？与以前的团队相比有什么区别?

9. 如果遇到战队成员无法解决的问题，你会寻求谁的帮助？你与其他团队或专家有多少沟通?

10. 你的微战队如何与其他微战队通信？对外接口人是否由单独的成员担任？他的具体职责是什么?

11. 你与战队中的其他人的交流频率是多少？每次沟通持续多长时间?

12. 你们工作中会出现分歧吗？通常是因为什么？频率大概是多少？最后是如何解决的？你认为这些分歧会如何影响战队的工作?

13. 你们微战队是否需要使用不同的技术实践？如果是这样，这些技术实践是如何被确定的？如果没有，那么微战队和你之前的团队有什么区别吗？

微战队工作与生活

14. 你们在工作以外的关系是什么？你们在工作之外有同样的爱好吗？

15. 你们是否会将工作带入日常生活？你认为别人呢？

16. 你们日常生活的情绪会影响你的工作吗？影响有多大？

开放问题

17. 哪些情况下，你认为不适合开展微战队实践？

18. 哪些情况下，你认为微战队可以发挥最好的效果？

B.2 第二次微战队访谈问题清单

被访谈战队背景

1. 用两三句话谈谈对微战队的理解。微战队是解决什么问题的？有什么好处？

2. 你们是由原来的团队转型的吗？是否有过有关微战队的培训？（如果不是，队长是如何选择的？）

3. 架构对战队组建的影响有多大？是软件架构的调整导致了微战队的产生，还是为了适应微战队的模式而调整了软件架构？你们战队是全功能的团队吗，是否是按照独立的特性团队组建的，还是几个微战队组成一个特性团队？

4. 战队里有多少人？你觉得影响战队人数的因素是什么？

5. 如果战队中有成员因为一些事情离开了战队，你们的应急计划是什么？你觉得什么情况会造成这种离队？

微战队工作流程

6. 队长主要负责管理工作吗？管理工作在平时工作中的比例是多少？能够全心全意的编码的时间一天中大约有多少？

7. 你们如何与其他团队或者部门交流的？战队内部又是怎么交流的？开发与测试时怎样交流的，与设计呢？

8. 你们团队间的交流成本下降了多少？主要体现在哪些方面？

9. 你们有晨会吗？晨会的模式是怎样的？

10. 你们和其他团队会遇到什么样的分歧？是怎么解决的？

11. 战队内部会遇到怎样的问题？是怎么解决的？

12. 你们战队的自组织度有多高？和以前相比呢？

13. 你们战队需要使用与原来不同的技术实践吗？比如看板。

开放问题

14. 哪些情况下，你认为不适合开展微战队实践？

15. 哪些情况下，你认为微战队可以发挥最好的效果？

16. 你们有哪些团建活动？

B.3 微战队问卷调查问题清单

战队的基本情况

1. 你们战队所属的部门是？

2. 你们负责开发的产品是什么？

3. 你们战队的人数有多少？

4. 你觉得影响战队人数的因素有哪些？

5. 你们战队是怎样组建的？

6. 你们战队有明确的职责（如开发、测试）分工吗？

7. 如果有明确的职责分工，你的职责是什么？

8. 你们的战队是全功能团队吗？

9. 同一个特性是由 1 个微战队完成的吗？

10. 你觉得微战队和架构之间的关系是怎样的？

11. 如果战队中有成员因为一些事情离开了战队，你们的应急计划是什么？

战队工作情况

12. 你是战队的队长吗？

13. 你觉得队长平时管理的工作占用了他多少时间？

14. 你每天全心全意编码的时间大约有多少？

15. 你与战队内成员主要通什么方式交流？

16. 工作外你与战队成员的联系密切吗?
17. 你与其他团队主要通过什么方式交流?
18. 你觉得交流成本下降主要体现在?
19. 你们在实践中使用过哪些敏捷方法?
20. 你们有晨会吗?
21. 你们解决战队内部的分歧最主要的手段是什么?
22. 你们如何解决与其他团队的分歧?
23. 你们在组建微战队后，运用了哪些以前没有用过的技术或流程?
24. 你觉得哪些情况适合使用微战队的模式去开发?
25. 什么情况又不适合去使用微战队的模式开发?

简历与科研成果

基本信息

黄璜，男，汉族，1996 年 7 月出生，江苏省宿迁人。

教育背景

2014 年 9 月 — 2018 年 6 月 南京大学软件学院 本科

攻读硕士学位期间完成的学术成果

1. **Huang Huang**, He Zhang, Xin Huang, Dong Shao, Chenxing Zhong, Xin Zhou, Jiali Chen, Shiqi Zhou, "DevOps and Microservices in Software Teams: A Cross-Company Ethnographic Study," in *Proc. ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering: Industry Track (ESEC/FSE 2021)*, Under Review
2. He Zhang, **Huang Huang**, Dong Shao, Xin Huang, "Fireteam: A Small-Team Development Practice in Industry," in *Proc. ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020)*, Nov. 2020.
3. **黄璜**, 张贺, 邵栋, "自动化工具对中国 DevOps 实践的影响," 软件学报, 2019, 30(10):3056-3070.
4. Nora Tomas, Jingyue Li, **Huang Huang**, "An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps," in *Proc. International Conference on Cyber Security and Protection of Digital Services (Cyber Security 2019)*, Jun. 2019.

攻读硕士学位期间参与的科研课题

1. 南京大学电信级微服务软件工程能力研究，2020–2021
2. 国家重点研发计划（政府间国际科技创新合作）项目：中挪联合面向供应链的高性能区块链系统支撑平台关键技术研究（2019YFE0105500），

2020–2021

3. 信息技术卓越教育与研究国际合作项目，2018–2021
4. 国家自然科学基金项目（面上项目）：基于软件资源库挖掘的过程仿真技术研究（61572251），2018–2019