A Cross-Company Ethnographic Study on Software Teams for DevOps and Microservices: Organization, Benefits, and Issues

Xin Zhou∗, Huang Huang†, He Zhang∗, Xin Huang∗, Dong Shao∗, Chenxin Zhong∗

∗State Key Laboratory of Novel Software Technology, Software Institute, Nanjing University, Nanjing, Jiangsu, China
†State Grid Nanjing Power Supply Company, Nanjing, Jiangsu, China

njuxinzhou,sgcc.huang.huang@gmail.com,hezhang@nju.edu.cn
njuhuangx@outlook.com,dongshao@nju.edu.cn, zhongcx@smail.nju.edu.cn

ABSTRACT
Context: DevOps and microservices are acknowledged to be important new paradigms to tackle contemporary software demands and provide capabilities for rapid and reliable software development. Industrial reports show that they are quickly adopted together in massive software companies. However, because of the technical and organizational requirements, many difficulties against efficient implementation of the both emerge in real software teams. Objectives: This study aims to discover the organization, benefits and issues of software teams using DevOps & microservices from an immersive perspective. Method: An ethnographic study was carried out in three companies with different business, size, products, customers, and degree of globalization. All the three companies claimed their adoption of DevOps and microservices. Seven months (cumulative) of participant observations and nine interviews with practitioners were conducted to collect the data of software teams related to DevOps and microservices. A cross-company empirical investigation using grounded theory was done by analyzing the archive data. Results: The virtual software teams were organized for adopting DevOps and microservices under the stubborn organizational structure. The adoption of DevOps and microservices brings benefits to rapid delivery, ability improvements and burden reduction, whilst the high cost and lack of practical guidance were emerged. Two major issues of adopting DevOps and microservices in software teams (i.e. fragmentary DevOps and abuse of microservices) were found common in the companies. Moreover, our observations and interviews reflect that in software teams, the relationship between DevOps and microservices is not significant, which differs from the relationship described in the previous studies. Four lessons for practitioners and four implications for researchers were discussed based on our findings. Conclusion: Our findings contribute to the understanding of the organization, benefits and issues of adopting DevOps and microservices from an immersive perspective of software teams.

CCS CONCEPTS
• Software and its engineering → Collaboration in software development.

KEYWORDS
DevOps, microservices, ethnographic study, participant observation, interview

ACM Reference Format:
Xin Zhou∗, Huang Huang†, He Zhang∗, Xin Huang∗, Dong Shao∗, Chenxin Zhong∗. 2022. A Cross-Company Ethnographic Study on Software Teams for DevOps and Microservices: Organization, Benefits, and Issues. In 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP ’22), May 21–29, 2022, Pittsburgh, PA, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3510457.3513054

1 INTRODUCTION
With the diverse and quickly changing user demands and the even fierce market competition, software organizations have to constantly improve the development process to deliver value quickly and implement more complex software systems. To address the problems arisen in the development process, DevOps, which was proposed from industry in 2009 [2] and originated from the combination of “development” and “operations”, offers a series of controllable and actionable Software Engineering (SE) strategies that can extend software design changes [3]. Moreover, DevOps emphasizes the emotional resonance and cross-stream collaboration within or between development teams and operation teams as well [11].

Emerging from Agile developer communities, microservices [14], which is the portmanteau of “micro” and “services”, was considered as the architecture solutions for continuous architecture setting in DevOps [5, 9, 43]. Microservices achieve benefits (e.g., on maintainability, reusability, scalability, availability and automated deployment) by decomposing the monolith into small services that communicate with each other through light weight mechanisms [24].

Both DevOps and microservices have received increasing attention from researchers and practitioners over the past decade [44]. Figure 1 shows Google search trends (by topic) for the two concepts since 2010 1, with a high degree of consistency. They are commonly adopted and implemented together in many software companies. Puppet’s report on DevOps [8] pointed out that microservices correlate with high performance under DevOps contexts while the report on microservices from O’Reilly [13] shows that more than half the companies with microservices utilize continuous deployment, the highest bar for automated DevOps.

1Accessed September 1, 2021.
The adoption and implementation of DevOps and microservices in many software teams while the public service in the product organizations. Specifically, the departmental barriers that still exist are found as the three main benefits in software teams, whilst iteration and deployment, enhanced ability and reduced burden are required. Under this circumstance, virtual software teams were evolved of DevOps and microservices in industry, and also contribute to the organization, whilst lacking the immersive thinking from software developers and teams, who are the atomic units of software development activities.

The objective of this ethnographic study is to provide an immersive perspective, revisiting software development activities as a member of software teams, to observe the adoption of DevOps and microservices in software teams and to discover the organization, benefits, and issues of software teams that adopt DevOps and microservices. The focus on software team’s daily work could offer an insight into the specific form of DevOps and microservices in real software development, which might be different from the original vision of the software organization, and also contribute to the practical implementation of DevOps and microservices. Specifically, activities of the teamwork from the three companies were observed in participant’s view as well as a series of open-ended interviews with practitioners in these companies were conducted to drive the investigation of the reciprocal impacts between software teams and the two concepts.

This study contributes to the understanding of software teams for DevOps and microservices by providing an immersive perspective. Organizational structure has become a barrier to the further evolution of DevOps and microservices in software organizations where strong organizational support and cultural accumulation are required. Under this circumstance, virtual software teams were organized for the adoption of DevOps and microservices. Rapid iteration and deployment, enhanced ability and reduced burden are found as the three main benefits in software teams, whilst high cost and lack of practical guidance were the two major pains. The adoption and implementation of DevOps and microservices in reality were found fallen short of the expectations of software organizations. Specifically, the departmental barriers that still exist in many software teams while the public service in the product expands over the course of business development. Moreover, the relationship between DevOps and microservices in software teams was found with weak significance, which reminds researchers and advocates of more attention to the real situation of practices in daily work.

2 BACKGROUND AND RELATED WORK

This section provides an overview of DevOps and microservices as well as some existing empirical studies focusing on their adoption and implementation in practice.

2.1 DevOps and Microservices in Industry

Despite both DevOps and microservices originated in the industry, there have been a number of related studies reported in academia, industry-orientation should be the original intention of the research on these two concepts.

Lwakatare et al. [27] conducted a multiple-case study in five companies with successful DevOps implementations, enhanced the definition of DevOps by emphasizing automation practices and confirmed the capabilities of DevOps for software development teams to deploy software in production environment fast. In addition to the technical practices, a supportive culture and mind-set, especially from senior management and customers that could effect cross-stream collaboration, was identified as critical in the long-term activity of the adoption and implementation of DevOps as well. Using grounded theory approach, Luz et al. [26] generated a theory to explain the successful DevOps adoption in fifteen companies and proposed a DevOps workflow, which was evaluated at a Brazilian Government institution. They found that the existing automation and tools can be enough for DevOps whilst collaboration is the core DevOps concern.

Zhang et al. [47] carried out a series of industrial interviews in thirteen companies to study the gaps between the visions and the reality as well as the benefits and sufferings of microservices around the nine characteristics claimed by Fowler and Lewis [14]. They claimed that organizational transformation, which is associated to the pains deserves further research directions. Bogner et al. [7] analyzed fourteen systems from ten companies with seventeen interviews, where most systems did not show the high degree of technological diversity as commonly expected with microservices. As the number of coarse-grained services continues to increase, the boundary line between service- and microservices-based systems is blurring.

Several studies focused on both DevOps and Microservices in industry settings. For example, Rademacher et al. [32] conducted two case studies to validate the applicability of the model-driven methodology, which uses viewpoint-based microservices modeling languages and respects DevOps teams’ concerns. The industrial case study conducted by Shahin et al. [38] showed that the characteristics of microservices meet the keys to DevOps success.

These empirical studies focus more on technical improvements to DevOps and microservices, and are more concerned with the overall characterization of DevOps and microservices in industry. They provide objective and rational descriptions of DevOps and microservices from a non-participant perspective. Although a number of interviews with practitioners and some observations...
on the practices were conducted in these studies, they are mostly carried with the researchers’ prior experience, thus ignoring some small but key influencing factors (e.g., personal experience at each software phase) in practice, especially from the human and social aspects. This ethnographic study increases the possibility of meeting important moments in the research process through the actual participation of researchers in the software teams, the basic unit of software development, to conduct participant observations and interviews. The interaction between software teams and DevOps together with microservices exposed in these processes could also contribute to the method improvement of DevOps and microservices [39].

2.2 Empirical Research on Software Teams
Recent research on software teams mainly focuses on those for agile methods, which emphasize the importance of teamwork [29]. High-quality teamwork can contribute to the success of projects in agile development, where the team leads and members have significant influences on the teamwork quality and performance (different on small and large projects) [25]. Team turnover of agile software teams was also studied. To better understand the change in the agile team, Hilton and Begal [19] conducted interviews and surveys in some software organizations with high turnover rates to explore the reasons why members left and joined. They further analyzed six-year data from employee database to distill the costs and benefits of the team’s turnover [19]. Researchers and practitioners were committed to exploring the organization of agile teams as well. Kniberg et al. [22] first introduced Spotify Model, a software development model driven by several autonomous squads. Salameh et al. [34, 35] used a variety of methods (e.g., case study and interview) to conduct a series of studies on the organization of this agile development team. Zhang et al. [45] analyzed the organization of fireteam, one agile development team, via their four key activities in software development. Similarly, this study focuses on agile software teams using DevOps & microservices to explore their organization, benefits, and issues from an immersive perspective.

3 RESEARCH METHODOLOGY
This section describes the research methodology step by step in this study. Following the guidelines [46], we conducted an ethnographic inquiry, which is used to study people and cultures [39]. Moreover, an ethnographic study offers insights into relationships between humans, process, technology and environment [46]. Aiming to discover particular groups of people’s perspectives in an organization [21], researchers endeavor to understand DevOps and microservices in software teams by investigating the daily reality of individuals or teams in various environments (e.g., the working space, communications, meetings and develop environments). The value of ethnography lies in its basis on pure observation, which is empirical in nature, and it is used to develop theories from an overall explanation [6]. Figure 2 shows the process of this study.

3.1 Research Questions
This study aims to explore the organization, benefits, and issues of software teams using DevOps & microservices in industrial contexts. With this objective, the following Research Questions (RQs) are formulated to guide this ethnographic study:

RQ1. How do companies organize software teams for DevOps and microservices?
RQ2. What are the pains and gains of DevOps and microservices to the teamwork?
RQ3. What issues do these software teams have when adopting DevOps and microservices?

3.2 Access to Ethnographic Fields
This research project began in January 2020 with a total of seven months of participant observations in the three companies at different scales. During the observations, the principal investigator (who took a leading role in this ethnographic study, including introducing the methods of observation and designing original semi-structured questionnaires for interviews) and the other authors conducted the participant observations in each company. This is a common ethnographic approach to immerse one or more researchers in the natural environment of the companies [10]. Table 1 shows the basic information about the three companies. They differ in business (domain), size, product, customer, and degree of globalization. All the three companies claimed to use DevOps and microservices. Therefore, they offer similar environments (DevOps and microservices) in different states of corporation development, which allows comprehensive information to be observed in the period of this study. This work follows the checklist proposed in [46]: multiple examples should be used while the individual field study is short in duration (far less than 8 months).
Table 1: Profiles of and interviewees from software organizations

| Information                      | Company 1 (C1)                                                                 | Company 2 (C2)                                                                 | Company 3 (C3)                                                                 |
|----------------------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Domain                           | Information technology & Telecommunication                                   | Big data & Artificial intelligence                                             | Software development                                                          |
| Size                              | 1000+ developers                                                               | 100-1000 developers                                                            | 50-100 developers                                                              |
| Customers                         | Infrastructure for information and communication technology                    | Databases and cloud platforms                                                  | Government and public utilities, colleges, financial institutions, energy companies, transportation companies, individual users |
| Globalization                     | Global enterprise                                                              | Domestic company, starting to go global                                        | Local company                                                                  |
| Experiences                       | DevOps (5 years), Microservices (6 years)                                      | DevOps (1 year), Microservices (5 years)                                       | DevOps (3 years), Microservices (5 years)                                      |
| Interviewees                     | Developer, software architect(4)                                              | Developer, (assistant) project manager(2)                                     | Developer/architect, product owner, CTO                                         |

Table 2: Observation items in the three companies

| Observation items           | C1                          | C2                          | C3                          | Description                                                                 |
|----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------------------------------------------------------|
| Company culture            | √                           | √                           | √                           | Company culture provides a great help for the investigators to familiarize themselves with the company and understand the reasons for some decisions. The history, brochures, training courses for new employees, and employees’ sense of identity with the company are focused on. Competitive products and the company’s living environment will also be considered in this observation item. |
| Working environment        | √                           | √                           | √                           | Working environment includes the working time, working spaces, afternoon tea and employee care in the company. Competitive products and the company’s living environment will also be considered in this observation item. |
| Communications             | √                           | √                           | √                           | Communication mainly appears between the investigators and other employees. Communication tools, contents, frequency, and efficiency are all concerned when observing. |
| Ongoing development processes | √                           | √                           | √                           | Ongoing development processes focused on the processes used in the software teams and they are commonly specified in the documents. Documents and engineer behavior are considered in conjunction with the observation. In addition, the researchers will actually participate in the processes. |
| Development environment     | ×                           | √                           | √                           | Development environment differs from the processes, pays more attention to some details such as language and tools. Some development resources (e.g., requirement documents, UI design drawings, source codes, operation reports) are also included in this observation items. |
| Meetings                   | ×                           | √                           | √                           | Meetings includes stand-up meetings, weekly meetings, business meetings and some other meetings. The topics discussed in the meeting, the organization format and the performance of the meeting participants will be observed. |

Two research students from the authorship applied for internships online or through company-college partnerships that offered them opportunities to join the teams and access the real contexts. After that, with the industry mentors’ help, the researchers understand the company culture and software team working process through various types of documents. During the internships, the researchers observed the practice status quo in processes of teamwork, which is elaborated in Section 3.3. They joined the software teams as team members and participated in their engineering activities, thus the research plan would not disturb the daily work of any other members.

3.3 Data Collection

Ethnography involves a variety of data collection techniques [12, 18], where the field study is the major method [36], and participant observations are most used in ethnographic field study in SE [46]. Interview is another important means to gathering data in ethnographic studies [17, 20]. Therefore, this ethnographic study combines the participant observations and interviews to achieve the triangulation of data collection, which is the basis of one ethnographic study and could help secure the quality of data and the validity of findings [12].

3.3.1 Participant observations. The participant observations were conducted in the three companies into three time periods between January and December 2020. First, the principal investigator (one of the research students) served as an intern product owner, which is a part-time job, in C3 from January to April 2020. Note that due to the outbreak of the Covid-19 pandemic, the company promoted working from home for about three months and the observation was carried out online during most of this period. Because of the assistance of remote collaborative office and meeting tools, it did not affect the observation. After that, the principal investigator was employed by C2 as a full-time assistant project manager from July to August. In December 2020, the principal investigator and the other research students joined C1 as the contributors of the company-college collaboration project to observe the practices related to DevOps and microservices. During the participant observations, the investigators worked with software teams, observed their daily activities as well as the behaviors of the team members and other employees, and shared insider’s views of what was happening in these organizations.

In general, the principal investigator and the other authors observed the DevOps and microservices related practices in these companies from three aspects of environment, workflow and technology. Table 2 shows some specific observation items in this study and the symbol ‘√’ stands for the observation item being observed in the company.

Note that because the research students got access into C1 through the company-college collaboration project, the observation of their environment was largely limited to the analysis of source code but the participation in their internal meetings was not allowed. This has caused some adverse effects (e.g., misunderstanding of some jokes) on our observations. Hence, a follow-up group interview (cf. Section 3.3.2) was conducted to observe the relationships among employees in C1.

3.3.2 Interviews. The interviews in C2 and C3 were conducted after the principal investigator left the companies, and the interviewees were selected from the team members whom the principal investigator ever worked with. The interviews in C1 were conducted together with the participant observation, and the interviewees in Table 1 were invited from the software teams collaborated in the college projects. Note that a group interview was carried out with a developer and three software architects from different teams (not limited to the collaborating team). The developers in C1 were interviewed twice. The developer interviewed in C3 is also responsible for some design decision-making at architecture level. Because of the small size of C3, the CTO is also involved in the daily development work as a team leader.

Driven by the RQs, the interview plan was jointly developed by all authors through brainstorming and meetings. For the setting of interview questions, we followed the principle that interviewees should elaborate more on their own views. As a result, a four-topic interview plan emerged:

**Topic1.** Job description and main responsibilities of interviewees.

**Topic2.** The organization of teamwork.

**Topic3.** Interviewees’ stories with DevOps.
**Topic 4. Interviewees’ stories with microservices.**

Topic 1 in C2 and C3 is used to find out whether the interviewees’ responsibilities ever changed since the principal investigator left the company. In C1, this topic is helpful because the interviewees came from different teams. Topic 2 discusses team organization and includes the impacts of their adopted practices on non-technical issues such as organizational structure and membership. Topic 3 and Topic 4 contain the interviewees’ motivations and journey to learn DevOps and microservices as well as their daily work on related team-department-organization. On the top two levels (department and company), this topic is helpful because the interviewees came from different teams. Topic 2 discusses team organization and includes the impacts of their adopted practices on non-technical issues such as organizational structure and membership. Topic 3 and Topic 4 contain the interviewees’ motivations and journey to learn DevOps and microservices as well as their daily work on related team-department-organization.

The discussion about these topics in interviews is open. “Let the interviewee say more” is our principle of conducting interviews. Because the interviewees are not in the exact same context, each interview might contain some unique topic-specific questions. For example, we asked what DevOps practices were used during the requirements analysis for project managers and product owners. These topic-specific questions were limited to the scope of software teamwork and the interviewees’ own experiences on DevOps and microservices, which could help us harvest immersive results from a comfortable interview atmosphere.

### 3.4 Data Analysis

Straussian Grounded Theory (GT) [41, 42], one of the methods to build theories and construct the reality in SE through interactions with language and communication [40], was employed as the data analysis technique in this ethnographic study since it provides a systematic method for generating conceptual theories from qualitative data sources [23]. Not starting from a preconceived conceptual framework, GT generates theories that evolve with the research process through the continuous interactions between the collected data and data analysis [1, 16, 41].

Open coding, axial coding and selective coding are the three major strategies in GT. Open coding is to identify the conceptual categories from the original data, which is the first step of GT. The code generated in this step is used to retain the exact words of the interviewees and the participant observations as much as possible. For instance, combined with the observed software development workflow (Figure 5), “in some internal product feedback online chat groups, software developers will also raise their discomfort when using the product, and will ask the product owner to improve the product for them” from C3 was labeled as “online group communication in C3” and “product feedback is not integrated in the pipeline in C3”. The observations on the ongoing development processes in C2 shows that the requirements management systems (JIRA) and code repositories (GitLab) are linked in the process, but software developers do not actually associate requirements with their own code in daily work. This was labeled as “separation of requirements and development in daily work in C2”.

Axial coding generates a set of new codes through comparing and merging the codes identified in the open coding phase. “Product feedback is not integrated in the pipeline in C3” and “separation of requirements and development in daily work in C2” were further coded as “the chasms in DevOps pipelines”.

After the tentative cores are found from the axial codes, selective coding is conducted to generate the theories following the cores. Around the tentative core “the chasms in DevOps pipelines”, the theory of “fragmentary DevOps” emerged.

Ethnography studies the practices as phenomena in the natural setting. In the processes of using GT for data analysis, we included some contexts (e.g., C3 in the label “online group communication in C3”) in the open codes, which might bring some difficulties (e.g., low level of abstraction) to axial coding. However, this could retain the characteristics of natural settings brought by ethnography.

### 3.5 Research Ethics

Although all the three companies we studied have cooperation agreements with us and approved our research on their development processes, we also take the responsibility of protecting the participating developers’ privacy. The research students were insiders of these teams and quietly followed others’ normal work style, which helps to protect the privacy of the companies and the developers being observed during the observation process. The interviews were conducted with the companies’ approvals as well, and all recordings, notes, and documents were handled properly to conform with the confidential agreements.

### 4 FINDINGS

This section presents the findings based on the participant observations and the interviews in this ethnographic study.

#### 4.1 Organization of Software Teams (RQ1)

**4.1.1 Stubborn organizational structure.** Although the three companies have different business settings, the organization of their software teams could be similarly abstracted as a three-level structure (Figure 3), which brings challenges to technology-driven changes and culture promotion.

![Figure 3: Three-level organizational structure](image)

**Visionary technology-driven organization.** The organizational structure is generally designed and documented by the senior management of the companies, that is, “commonly technical-free”. Other considerations and constraints may come from human resources, finance, and other departments. When the departments attempted internal trials on shifting software teams, the decision makers would consider the ideas and interests of all team members as well as other departments. This process may last a long period and is likely to be rejected, for example in C1, “many department heads do not have the gumption to persuade decision makers”. In Figure 3, technology-driven innovation requires a four-level chain of group-team-department-organization. On the top two levels (department
and organization), "technology is often not the top concern and option to solve the problem".

Deferred culture switch. Trust, the cultural core of DevOps and microservices, needs to be cultivated across software teams. Complaints are still the most prominent collaboration problem among cross-stream collaborations in the three companies. Moreover, some developers are resistant to changes, which would cause extra efforts on them. “Developers go into development with a mindset of how they used to develop” and the mindset would take a long time to change. Although knowledge-sharing has become a consensus, it takes time to be fostered in software teams. After a knowledge-sharing course on database in C2, merely less than 10% of the participants expressed their opinion about the content (Figure 4), which may indicate that the current climate of knowledge-sharing is not as expected yet.

Figure 4: A conversation after a knowledge sharing in C2

4.1.2 Virtual software teams. Affected by the stubborn organizational structure, when using DevOps and microservices for software development, the software team may work as a virtual team.

Figure 5 shows the workflow of a typical virtual software team for C3. Each microservice is maintained by the virtual team composed of engineers, who “may take on tasks in different development teams”, from five teams (i.e. product, UI design, development, test and operations&sale team), which also “achieve the combination of development and operations”. The development tools for the virtual team were not unified. In C3, self-developed platform was used for requirement development while design-sharing platform (Lanhu) and GitHub were used for UI and code management. Similarly, C2 uses JIRA and GitLab respectively for requirements and code management. Virtual teams do not work physical together. They use meetings (e.g., requirements review meeting, code review meeting) to solve problems encountered in understanding microservices and DevOps. An extreme situation experienced was that developers participated in different virtual team meetings throughout the day, leaving no time for development.

4.2 Gains and Pains of Adopting DevOps and Microservice (RQ2)

4.2.1 Gains. Three benefits were found to be significant in observations and were widely mentioned in interviews.

Rapid iteration and deployment. Rapid iteration and deployment are the major benefits from the perspective of the organizations, which allow them to create value to market quickly. Both DevOps and microservices contribute to these benefits. With the adoption of DevOps, “the organization sets the rules, builds the basic pipeline, and provides the automatic tools”. Meanwhile, the adoption of microservices “brings new tools and methods, e.g., containers and cloud native to software teams”. Software teams adopt new techniques and methods to solve problems during implementation of microservices, and the outcomes then will be fed back to the organization. C3, as a startup, employs DevOps and microservices to quickly and consistently deliver on-demand features and release updates once or twice a week. In C2, more than 70% of teams implement continuous integration on a daily basis, some of them are able to run more than five pipelines a day on average.

Enhanced ability and skills. This can be recognized as the main benefit for individuals that “could help them to consider new job opportunities”. It results from requirements that a single team has to be capable for almost full spectrum of responsibilities in the entire life cycle of DevOps. Another possible reason is that team turnover is likely to be higher in DevOps. Individual’s enthusiasm for learning might be stimulated. One interviewee from C1 stated that “it is great for engineers to change teams in terms of their development tasks”.

Reduced burden. This benefit can be attributed to microservices: “microservices allow developers to focus on specific domains”. It further helps “increase the happiness of software teams”. However, the benefit “enhanced ability and skills” of DevOps, which requires developers to take on more tasks, may has a negative impact on this.

4.2.2 Pains. Two major pains while adopting DevOps and microservice were distilled in this study.

High cost. This might be the most significant pain brought by both DevOps and microservices, similar to the problem arising whenever a new technology or method is adopted. C3 took a full year to replace 95% of the implementations of a particular technology for mobile in the pipeline. The consumption of more resources of microservices results in the high cost as well. The shortage of hardware resources, especially for testing, was commonly mentioned by microservices results in the high cost as well. The shortage of hardware resources, especially for testing, was commonly mentioned by microservices. The deployment costs for the virtual team in C3, as a startup, employs DevOps and microservices to quickly and consistently deliver on-demand features and release updates once or twice a week. In C2, more than 70% of teams implement continuous integration on a daily basis, some of them are able to run more than five pipelines a day on average.

Lack of practice guidelines. This pain is especially significant in microservices-oriented decomposition. Although many approaches (e.g., Domain-Driven Design and dataflow-based design) for microservices decomposition have been proposed, software teams occasionally felt overwhelmed in their daily work, where the business changes seem out of their control. Along with the business evolution in C1, many microservices began to merge back to larger ones, which become no longer ‘micro’ services. C3 showed that when a new business is created, it is difficult and complex to split some new domains or reuse the old domains.
4.3 Issues of Using DevOps and Microservices (RQ3)

4.3.1 Fragmentary DevOps. Both the above gains and pains are also reported in the previous studies (e.g., [33, 37]), which suggests that DevOps implementations we observed at the three companies to some extent were common to others. However, some reported benefits such as “frequent releases help to reduce stress levels” [33] were not directly observed in this ethnographic study, which might be caused by fragmentary DevOps.

Figure 5: Observed workflow of a virtual team in C3

![Figure 5: Observed workflow of a virtual team in C3](image)

Figure 6: Two chasms in DevOps pipelines

First, software organizations are keen to improve DevOps pipelines (automation, security, etc.), but may have less interest in other essentials (e.g., cross-stream communications) that DevOps advocates for the holistic improvement. The complete DevOps pipeline was detached into several segments that are barely connected to each other (Figure 6). The first chasm is between planning and coding. Although C2 builds an association between JIRA and GitLab, the correlation between requirements and code in the pipeline is elusive. The self-developed project management system in C3 has little to no connection with the code repository. Another chasm is between operation and development. For example, operational issues could only be resolved in weekly operations meetings in C3. Moreover, the operations of the Internet infrastructure produced in C1 and C2 are independent from their development. This chasm raises questions about whether DevOps is indeed adopted in organizations because of the separation of 'Dev' and 'Ops'. Some software teams do not need to use the complete DevOps pipeline in their daily work. Consequently, they only care about the portion of the pipeline that they are involved in. The observed workflow in C3 (Figure 5) shows the sense of fragmentation between development activities.

Second, departmental barriers still exist, in some cases deeper than ever. Given the uncertainty of the assignment in some cross-stream tasks, software teams tend to meet their own interests in development. One developer in C1 admitted that “he wanted other teams to fork the code instead of asking for more functional requirements”. The activity of Test and Verification in C3 is the most controversial in the entire process because of the involvement of four different teams. Most of the reasons for postponing the release come from this activity. On the other hand, in the observed three companies, although there exist knowledge-sharing platforms (e.g., public lectures), software teams preferred to share technologies and methods within their own team rather across teams, which might complicate the learning process for other teams.

All the three companies claim that they have adopted and implemented DevOps. However, the observed adoption of DevOps practices is fragmentary. This calls the need for DevOps maturity, which offers a clearer understanding of the practice levels of software teams and guides the successful adoption of DevOps in software organizations.

Figure 7: Dependencies of microservices in C1

![Figure 7: Dependencies of microservices in C1](image)
4.3.2 Abuse of microservices. When re-thinking of the pains caused by microservices, we found that the abuse of microservices might be the major root cause, where “blind pursuit of fancy technologies” and “excessive agility in software development” play the important roles.

Microservices may not match all circumstances. The architects in C1 confessed that “the implementation of microservices in telecom context was not a good choice for them at the current stage”. Figure 7 shows the dependencies of some microservices in one project in C1. The ingoing and outgoing dependencies of m4, m7 and m17 are significantly overloaded that architects spontaneously merge the mistakenly decomposed microservices. Some employees in C2 preferred to SOA rather microservices advocated by the company.

C3 claimed that “microservices could now support their business development, but when the requirements accumulate, many problems may arise”. It might be cool for software organizations or teams to upgrade to new technologies, whereas ignorance of reality, especially business, is likely to be counterproductive.

5 DISCUSSION

This section discusses the relationship between DevOps and microservices in software teams, followed by discussing the lessons learned based on the findings.

5.1 Relationships between DevOps and Microservices

5.1.1 In related work. Researchers considered DevOps and microservices interrelated since their inception. Enabling effective DevOps implementation by increasing the importance of small teams [5], microservices are considered as the enabling architecture style for continuous delivery and DevOps [9, 43]. Waseem et al. identified 47 primary studies in their mapping study on microservices in DevOps [44], including development and operations of microservices (e.g., [31]), approaches and tool support for microservices-based systems (e.g., [28, 30]), and experiences in migration toward microservices (e.g., [4, 5]), which form the three major themes to describe microservices in DevOps. Figure 8 shows the classification summarized in their study. They also collected fifty tools in support of microservices in DevOps. Employing microservices in DevOps was observed to have positive effects on most quality attributes.

Figure 8: Classification of research themes on microservices in DevOps [44]

5.1.2 In our observations and interviews. One relationship in [44] (the development and operations of microservices in DevOps) can be observed from the three companies in this study, whereas the other two were reflected as a weak relationship (shown in Figure 9).

No interviewee voluntarily mentioned the connection between DevOps and microservices. For many developers in the three companies, the connection between the both is limited to “microservice systems that they develop were running on the DevOps pipelines”. The DevOps platform could support the development with or without microservices, meanwhile microservices could be deployed and released in other ways than DevOps pipelines. The software teams and developers are more concerned with getting their daily work done and delivering the project on time than setting up the connection between them, which “are most likely to be mentioned in knowledge-sharing sessions”.

From the perspective of software developers, all the approaches and tools as well as the experiences from migrating to microservices should serve the product development. Consequently, in software teams’ daily work, the three aspects reported in [44] could be integrated together as developing microservices on DevOps pipelines. For example, C2 and C3 use Docker to “deploy microservices so that the consistency of the development and product environments in DevOps pipelines could be guaranteed to a certain extent”.

The lack of joint training on DevOps and microservices could be one reason for the weak correlation. It is observed that although all the three companies offer technical training for their software teams (e.g., cooperative courses with colleges in C1 and enterprise’s college’ in C2) and many well-known companies have proposed solutions for microservices with DevOps on Cloud, no courses for the integration of DevOps and microservices in these companies. Moreover, developers are more interested in the technique-specific training courses (e.g., Kafka) and experience courses (e.g., innovative AI solutions for securities industry).

Another significant reason for the disconnection is that opinion leaders lack sufficient knowledge. The department heads or the team leaders, who could be the opinion leaders in software teams, would affect other team members’ understanding of such practices as well as the team’s development processes. Although they might participate in some courses, their knowledge of the practices is limited. For example, one leader in C3 has recognized that “he does not know much about DevOps”. In this case, the participant observation from researchers may trigger the software teams’ awareness of the concepts, thereby improving related practices in teamwork, e.g., security has become a bond between microservices and DevOps in C2. On the other hand, the appeal of researchers is also crucial for the further evolution of practices, e.g., monitoring, security, and performance degradation issues in [44].
5.2 Lessons Learned

Section 4 reports the organization of software teams that adopt DevOps and microservices, as well as the gains, pains, and issues of DevOps and microservices. The lessons for practitioners and the implications for researchers were distilled from four aspects (organization, DevOps, microservices, and their relationship).

5.2.1 For practitioners.

Global perspective. Since the stubborn organizational structure becomes the obstacle against technology-driven changes and culture shift (cf. Section 4.1), practice innovation in software teams needs to address the interests on the organization level (not single software team) to seek senior management’s support to achieve successful implementation.

More connections. To tackle the fragmentary DevOps (cf. Section 4.3.1), it is essential for software teams to establish the connections between requirements (plan), prototype (design), and code, as well as take the operations on systems into account in development.

Active adaption. To mitigate the abuse of microservice (cf. Section 4.3.2), practitioners should adapt the architecture to match their domain model, e.g., refactoring microservices and choosing other architectural styles.

Joint training. Training on DevOps and microservices together helps practitioners better understand the tasks distributions, thus supports the development in software teams.

5.2.2 For researchers.

Organization structure. Since the issues of adopting DevOps and microservices come from organization structure (cf. Section 4.3), it is worth studying the timing of adjusting organization structure and the methods of avoiding new structures solidification.

Maturity model. Although the three companies claimed that they all adopted DevOps, they may have divergent understanding and practice of DevOps, which calls for joint force of academia-industry collaboration to develop the DevOps maturity model.

Architecture migration. The migration to microservices has become a popular research topic, where the risk of the abuse of microservices also needs to investigate. Further, the migration to next-generation architecture (e.g., microservices to serverless architecture) is worth exploring by researchers.

Practices-oriented courses. The development of a joint training framework (practices-oriented courses) for DevOps and microservices will involve both researchers and practitioners.

6 THREATS TO VALIDITY

Internal validity. Reflexivity is very sensitive to ethnologists because of its impacts on the understanding and description of the observed practices [6]. The personal experiences as well as social and cultural background may become inherent biases in ethnographic research, which would be reflected in the participant observations. To alleviate this bias, the principal investigator had meetings for discussion in the process of data analysis with other authors who did not participate in all the observations. The raw data by specific engineers (both observation and interview) and codes generated during data analysis are inconvenient to be disclosed in the paper due to corporate confidentiality policies. Specifically, the principal investigator describes the observed phenomena and contexts, and then other authors expressed their opinions on these observations from their own perspectives. This constitutes a virtual non-participant observation process. Meanwhile, in order to describe the daily work of software teams, the perspectives of the principal investigator and other authors should also be involved in this study.

External validity. Although the studied three companies come from different domains and offer rich data on DevOps and microservices in software teams from participant observations and interviews, we do not claim that our results are valid for other scenarios for the reason that these companies share a similar cultural background. Whereas some findings from this study confirm the findings of other studies in different contexts. As this study aims at offering insights into DevOps and microservices in software teams from an immersive perspective, software organizations in similar context could benefit from our findings while other companies could come up with their own findings with reference to our results.

7 CONCLUSIONS

This paper reports our ethnographic study that combines participant observations and interviews to investigate the adoption and effects of DevOps and microservices in real software teams. We observed three companies from company culture, working environment, communications, ongoing development processes, development environment, and meetings. Eight individual interviews and one group interview were conducted to achieve the data triangulation.

The overall adoption and implementation of DevOps and microservices (e.g., benefits and challenges) by all the three companies is consistent with industry practices reported in previous studies. Since the stubborn organizational structure remains the obstacle against the technology-driven organization and culture switch, virtual software teams are popular in the organizations adopting DevOps and microservices. The three main gains (i.e. rapid iteration and deployment, enhanced ability, and reduced burden) and two major pains (i.e. high cost and lack of practical guidance) are found from the observations and interviews. Furthermore, fragmentary DevOps from the aspects of pipelines and department barriers and abuse of microservices caused by herd mentality and excessive agility are identified as the two issues of DevOps and microservices in software teamwork. We also discuss the weak correlation between DevOps and microservices in software teams. Furthermore, four lessons for practitioners and four implications for researchers are distilled from the findings to implement and study DevOps & microservices in software teams.

Our ethnographic study contributes to the understanding of software teams that adopt DevOps and microservices from an immersive perspective, which helps understand the interactions between software teams and DevOps & microservices. We encourage more studies with immersive observations on DevOps and microservices to emerge, so that the adoption and impacts of new technologies and practices in software teams would become traceable, which further enables comparative studies on DevOps and microservices.
ACKNOWLEDGMENTS
This work is supported by the National Natural Science Foundation of China (No.62072227), the National Key Research and Development Program of China (No.2019YFE0105500) jointly with the Research Council of Norway (No.309494), the Key Research and Development Program of Jiangsu Province (No.BE2021002-2), Intergovernmental Bilateral Innovation Project of Jiangsu Province (No.BZ2020017), as well as the Scientific Research and Innovation Program of Jiangsu Province (KYCX-21-0061).

REFERENCES
[1] Steve Adolph, Wendy Hall, and Philippe Kruchten. 2011. Using Grounded Theory to Study the Experience of Software Development. Empirical Software Engineering 16, 4 (2011), 487–513.
[2] John Allspaw and Paul Hammond. 2009. 10+ Deploys per Day: Dev and Ops Cooperation at Flickr. In: Velocity: Web Performance and Operations Conference.
[3] Matey Artac, Tadej Borovsak, Elisabetta Di Nitto, Michele Guerrero, and Damian Andrew Tamburri. 2017. DevOps: Introducing Infrastructure-as-a-Code. In Proceedings of the 9th International Conference on Software Engineering: Companion. IEEE, 497–498.
[4] Amrin Balalaei, Abbas Heydarnoori, and Pooyan Jamshidi. 2015. Migrating to Cloud-Native Architectures Using Microservices: An Expo Primer. In Proceedings of the 4th European Conference on Service-Oriented and Cloud Computing, Springer, 201–215.
[5] Amrin Balalaei, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices: Architecture Enabler for DevOps: Migration to A Cloud-Native Architecture. IEEE Software 33, 3 (2016), 42–52.
[6] Simon Elias Bibri. 2019. Awareness Aspects and Effects on Coordination and Information Sharing in the Domain of Software Engineering as CSCIW. An Ethnographic Study. Augmented Human Research 4, 1 (2019), 1–20.
[7] Justus Bogner, Jonas Fritsch, Stefan Wagner, and Alfred Zimmermann. 2019. Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality. In Proceedings of the 16th International Conference on Software Architecture: Companion. IEEE, 187–195.
[8] Alanna Brown, Michael Stahinke, and Nigel Kersten. 2020. State of DevOps 2020. Technical Report. Puppet.
[9] Lianping Chen. 2018. Microservices: Architecting for Continuous Delivery and DevOps. In Proceedings of the 15th International Conference on Software Architecture. IEEE, 39–46.
[10] John W. Creswell and J. David Creswell. 2017. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. Sage Publication.
[11] Andrej Dyck, Ralf Penners, and Horst Lichter. 2015. Towards Definitions for Release Engineering and DevOps. In Proceedings of the 3rd International Workshop on Release Engineering. IEEE, 3–8.
[12] David M. Fetterman. 2009. Ethnography: Step-by-Step. Sage Publication.
[13] Neal Ford. 2020. The State of Microservices Maturity. Technical Report. O'Reilly.
[14] Martin Fowler and James Lewis. 2014. Microservices: A Definition of This New Architectural Term. https://martinfowler.com/articles/microservices.html Accessed February 1, 2021.
[15] Jonas Fritsch, Justus Bogner, Stefan Wagner, and Alfred Zimmermann. 2019. Microservices Migration in Industry: Intentions, Strategies, and Challenges. In Proceedings of the 35th International Conference on Software Maintenance and Evolution. IEEE, 481–490.
[16] Barney G. Glaser and Anselm L. Strauss. 2009. The Discovery of Grounded Theory: Strategies for Qualitative Research. Transaction Publishers.
[17] Martyn Hammersley. 2018. What Is Ethnography? Can It Survive? Should It? Ethnography and Education 13, 1 (2018), 1–17.
[18] Martyn Hammersley and Paul Atkinson. 2007. Ethnography: Principles in Practice. Routledge.
[19] Michael Hilton and Andrew Begel. 2018. A study of the organizational dynamics of software teams. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice. ACM, 191–200.
[20] John A. Hughes. 1995. Ethnography, Plans and Software Engineering. (1995).
[21] Paul Johannesson and Erik Persson. 2012. Creating a code base around制品. Information and Software Technology 56, 2 (2014), 252–271.
[22] Cheng Zhang, He Zhang, Zijia Jia, Cheng Zhang, Jiaqi Li, Quyua Gao, and Zhiqun Shen. 2019. A Data-Driven Flow to Identify Microservices from Monolithic Applications. Journal of Systems and Software 157 (2019), 110380.
[23] Yngve Lindsjørn, Gunnar R. Bergersen, Torgeir Dingsøyr, and Dag IK Sjøberg. 2018. Teamwork quality and team performance: Exploring differences between small and large agile projects. In Proceedings of the International Conference on Agile Software Development. Springer, 267–274.
[24] Walter Pinheiro Luz, Gustavo Pinto, and Rodrigo Bonfáció. 2019. Adopting DevOps in the Real World: A Theory, A Model, and A Case Study. Journal of Systems and Software 157 (2019), 110384.
[25] Lucy Ellen Lwakateki, Terhi Kilamo, Teemu Karvonén, Tanja Sauvala, Ville Heikkin, Juha Ikonen, Pasi Kuvaja, Tommi Mikkonen, Markku Ovso, and Casper Lassenius. 2019. DevOps in Practice: A Multiple Case Study of Five Companies. Information and Software Technology 114 (2019), 217–230.
[26] Marco Miglierina and Damian A Tamburri. 2017. Towards Omnía: A Monitoring Factory for Quality-Aware DevOps. In Proceedings of the 8th International Conference on Performance Engineering: Companion. ACM, 145–150.
[27] Dinesh Neur, Badal Kumar Jha. 2005. Challenges of migrating to agile methodologies. Commun. ACM 48, 5 (2005), 72–78.
[28] Tony Salvador, Genevieve Bell, and Ken Anderson. 1999. Design Ethnography. Design Management Journal (Former Series) 10, 4 (1999), 35–41.
[29] Abdallah Salameh and Julian Blass. 2018. Influential factors of aligning spotlight squads in mission-critical and offshore projects—a longitudinal embedded case study. In International Conference on Product-Focused Software Process Improvement. Springer, 199–215.
[30] Abdallah Salameh and Julian M Blass. 2019. Spotify tailoring for promoting effectiveness in cross-functional autonomous squads. In International Conference on Agile Software Development. Springer, 29–39.
[31] Tianfu Tao, Jinghua Luan, and Guangzhou Yang. 2018. Empirical Study on Agile Development Teams and Enterprise Scrum. In Proceedings of the 2018 IEEE 23rd International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 175–184.
[32] Helen Sharp, Yvonne Dittrich, and Cedric CB De Souza. 2016. The Role of Ethnographic Studies in Empirical Software Engineering. IEEE Transactions on Software Engineering 42, 6 (2016), 786–804.
[33] Dag I.K. Sjøberg, Tore Dyba, Bente C.D. Andia, and Jo E. Hammy. 2008. Building Theories in Software Engineering. In Guide to Advanced Empirical Software Engineering. Springer, 312–336.
[34] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. In Proceedings of the 30th International Conference on Software Engineering. ACM, 120–131.
[35] Anselm Strauss and Juliet Corbin. 1998. Basics of Qualitative Research Techniques. Citeseer.
[36] Davide Tabi, Valentina Lenarduzzi, and Claus Pahl. 2018. Continuous Architecting with Microservices and DevOps: A Systematic Mapping Study. In Proceedings of the 8th International Conference on Cloud Computing and Services Engineering. Springer, 126–151.
[37] Muhammad Waseem, Peng Liang, and Mojtahab Shahin. 2020. A Systematic Mapping Study on Microservices Architecture in DevOps Transformation: An Industrial Case Study. In Proceedings of the 35th International Conference on Software Engineering: Companion. Springer, 57–67.
[38] Zahid Ahmad, Muhammad Ali Babar. 2020. On the Role of Software Architecture in DevOps Transformation: An Industrial Case Study. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, ACM, 57–67.
[39] Abdallah Salameh and Muhammad Ali Babar. 2020. On the Role of Software Architecture in DevOps Transformation: An Industrial Case Study. In Proceedings of the 39th International Conference on Software Engineering: Companion. ACM, 1365–1375.
[40] Lasse Heikkilä, Juha Itkonen, Pasi Kuvaja, Tommi Mikkonen, Markku Ovso, and Casper Lassenius. 2019. DevOps in Practice: A Multiple Case Study of Five Companies. Information and Software Technology 114 (2019), 217–230.
[41] John Allspaw and Paul Hammond. 2009. 10+ Deploys per Day: Dev and Ops Cooperation at Flickr. In: Velocity: Web Performance and Operations Conference.
[42] Paul Johannesson and Erik Persson. 2012. Creating a code base around制品. Information and Software Technology 56, 2 (2014), 252–271.
[43] Cheng Zhang, He Zhang, Zijia Jia, Cheng Zhang, Jiaqi Li, Quyua Gao, and Zhiqun Shen. 2019. A Data-Driven Flow to Identify Microservices from Monolithic Applications. Journal of Systems and Software 157 (2019), 110380.