Microservice Architecture in Reality: An Industrial Inquiry

He Zhang, Shanshan Li, Cheng Zhang[†], Zijia Jia, and Chenxing Zhong
State Key Laboratory of Novel Software Technology, Software Institute, Nanjing University, Jiangsu, China
[†]School of Computer Science & Technology, Anhui University, Anhui, China
Email: {dr.hezhang, stu.shanshan.li}@gmail.com, cheng.zhang@ahu.edu.cn, {stu.zijiajia, chenxingzhong1125}@gmail.com

***Abstract*—*Background:* Seeking an appropriate architecture for a software design is always a challenge in recent decades. Although microservices as a lightweight architecture style is claimed that can improve the current practices with several characteristics, many practices are based upon the different circumstances and reflect the variant effects. An empirical inquiry brings us a systematic insight into the industrial practices on microservices. *Objective:* This study is to investigate the gap between the ideal visions and real industrial practices on microservices and what benefits we can gain from the industrial experiences. *Method:* We carried out a series of industrial interviews with thirteen different types of companies. The collected data were then codified according to the defined qualitative methods. *Results:* We characterized the gaps between the typical characteristics accepted in the community and the industrial practices of microservices. Furthermore, the compromise between benefits and sufferings of microservices around these nine dimensions were also investigated. *Conclusion:* We confirmed the benefits of the microservices that can be obtained from practice as well as their possible pains that need to be addressed with extra expense from experiences. Besides, some outlined pains, e.g., organizational transformation, decomposition, distributed monitoring, and bug localization, may inspire researchers to conduct the further research.**

***Keywords*—microservices, empirical study, interview, pains**

I. Introduction

Microservices Architecture (MSA) has become the latest trend in software development. It advocates the concept of componentization, based on which a single application is implemented as a set of small and independent services [1]. Each service of MSA runs in its own process and communicates with others through lightweight mechanisms [2]. Many advantages related to MSA have been justified both in academia and industry, for example, scalability and independent deployment. Inspired by the perceived benefits of MSA, a large amount of world-leading Internet services such as Netflix, Amazon, and eBay have migrated to MSA.

However, research and practice in recent years justify the statements of Fowler and Lewis [2] that MSA is not a silver bullet and it has several challenges in different phases [3]. Due to the distributed setting of services, MSA could make the development and operation of microservices more complicated in practice, and lead to potential problems in data consistency, transaction management and service communication within the network [4], [5]. As a consequence, it is important for both industry and academia to shed light on the gaps between the ideal characteristics and the realistic practices of MSA, and understand the trade-offs among the benefits as well as the costs coming with [6]. First, the widely-concerned characteristics and common industrial views of MSA in different domains can aid the practitioners especially the newcomers in migrating to MSA more efficiently and successfully. The benefits and accompanying pains may impact the decision-making about migrating legacy systems to MSA or developing a new system using MSA style. Moreover, pains encountered by practitioners may inspire the researchers who are willing to identify potential research opportunities in this area.

There are several studies that investigated the state-of-the-art of MSA adoption [4], [7], [8], the processes and the activities of MSA migration [9], [10]. To the best of our knowledge, none of them provided an in-depth gap analysis between the ideal characteristics to be expected for MSA and the reality from practical experiences with MSA in different domains. Furthermore, the compromise between the benefits and the sufferings of MSA in practical context does not gain enough attention of the relevant studies.

Motivated by the above absent knowledge in the recent studies about MSA, we intended to not only clarify the gaps between the common visions and the practical state of MSA, but also understand the trade-offs between its recognized characteristics and its accompanying pains when designing, implementing and operating the microservices. To achieve this, we conducted a series of semi-structured interviews with 13 different types of companies. The data collected from the participants contributes to reveal the state-of-the-practice, including the concerns, benefits, and pains along with in MSA.

The main contributions of this study are threefold: i) It identifies the gaps between the common characteristics and the real practices on MSA; ii) It provides the trade-off analysis of the benefits and the pains of MSA; and iii) It discusses the potential future research directions in this area from both the practitioners' and the researchers' perspectives.

The remainder of this paper is structured as follows. Section II explains the research method of this study. Section III shows demographic results of all participants. In section IV, we report the findings that answer our research questions. Section V discusses some problems and potential research directions identified in our research. Section VI describes threats to validity of our study. Section VII presents an overview of related works. Section VIII draws the conclusion and suggests the future work.

978-1-7281-0528-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSA.2019.00014

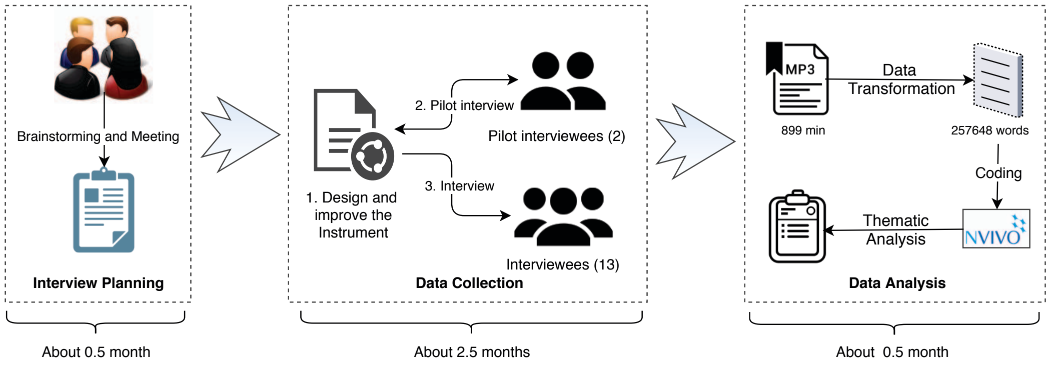

Fig. 1. The procedure of the interview study.

II. Research Method

This study was initiated in the second half of 2018 and then carried out for four months using a common qualitative research approach–interview [11]. Three student researchers and their two supervisors participated in designing the interview instrument, collecting the data through interviewing the practitioners, and analyzing the data for reporting.

A. Research Questions

With the objective of characterizing the practices and challenges related to the migration and adoption towards MSA, we defined the following two research questions (RQs):

RQ1: What are the gaps between the visions and the reality of microservices? This RQ aims to investigate the real practices in terms of the claimed specific characteristics of microservices.

RQ2: What are the benefits and the expenses of implementing microservices in practice? This RQ is to identify the possible gaps between the positive impacts of microservices in theory and the perceived pains in industrial practices, which may further drive the future research directions.

We took the best known visions in the community, which was proposed by Fowler and Lewis in 2014 [2], as the reference of the characteristics of microservices researched in this study.

B. Research Procedure

This study is aimed at gaining an in-depth understanding of the state-of-the-practices of MSA in software industry, with specific focus on exploring the gaps between the common visions and the real practices on microservices. We therefore collected the data from practitioners with experiences in MSA using semi-structured interviews, which is a common research method for collecting qualitative data [12]. The interviews were conducted through a high degree of discussion between interviewers and interviewees with a mixture of open-ended and specific questions, which enable the researchers to obtain foreseeable and unexpected data across different interviews.

The procedure of this interview-based inquiry is illustrated in Fig. 1. At the initial stage, three student researchers did the brainstorming to jointly identify the interests of MSA and had the planning meeting with their supervisors to determine the plan of the interview. Then they designed the interview instrument with respect to the interests obtained from the meetings as well as inspired by the nine characteristics claim by Fowler and Lewis [2] and other previous MSA related survey studies [9], [10], [13]. Meanwhile, the researchers started to invite suitable practitioners to participate the interviews. We searched the experienced practitioners among the speakers in three influential industrial forums related to software architecture in China. We then invited the selected practitioners who worked in the domain of software architecture for more than two years and also had the experience on MSA as our interviewee candidates. Finally, the practitioners from thirteen companies accepted our invitations. To evaluate and improve the interview instruments, two pilot interviews with the carefully selected domain experts from our partners were rigorously carried out before conducting interviews with the invited interviewees. The instruments experienced eight revisions based on the useful feedbacks from the two pilot interviews.

C. Data Collection

The instruments of our semi-structured interviews was designed with the reference to the guide by Kvale [11], and reviewed by the two supervisors with rich experiences in interview studies. During the pilot interviews with two domain experts, the structure was improved and some questions were clarified. The final version of our interview instrument is divided into three parts:

1) Warm-up: Basic information about the practitioners and their companies;
2) Practices of MSA: Concerns and practices around the characteristics of microservices claimed by Fowler and Lewis [2];
3) Gains and Pains: Benefits and sufferings of microservices, and the ones that need further research.

Some questions in the semi-structured interviews are designed to be straightforward and closed-ended, e.g., the questions for the basic information. While others are open-ended to steer the interview under different circumstances, which means that the order or the details of certain questions are not fixed somehow depending on the interviewee's experience and answers.

This interviews were mostly conducted by face-to-face the participants' sites located in five cities in China. Only one interview was conducted online due to the geographic restriction. We spent about 2 months on carrying out all thirteen interviews. Each interview involved one interviewee and two or three researchers as the interviewers. During the interview process, one researcher was responsible for asking the questions with the guide of the instruments, while the other(s) took notes and posed complementary questions to ensure the interest was covered. All interviews took 15 hours in total with an average of 69 minutes each. Meanwhile, in case of losing some important information, all the interviews were recorded using a digital voice recorder after getting the interviewees' permissions.

D. Data Analysis

The data collected from the interviewees was processed and analyzed to generate the key findings of this study. The transcription of interview recordings was first done by one student researcher with the aid of a speech recognition application and then checked by the other two student researchers. The transcripts of the 15 hours' interviews that contain 257648 words were coded with the notes token during interviews by following the coding manual [14]. To avoid introducing researcher's bias, the coding of each interview transcripts was done by three student researchers independently. Any disagreements were discussed and solved through regular meetings with their supervisors.

As recommended in [15], the data analysis in this study combined both manual and computer-assisted methods with the facility by a qualitative data analysis package–NVivo, which is able to assist researchers in efficiently exploring the patterns from large amounts of textual data and improve the rigor of the analysis process. Nevertheless, because NVivo is less useful in terms of thematic ideas generation, we applied the thematic analysis [16] that manually analyzed the concerns and practices in different phases of MSA.

III. Demographic Results

The participants to this interview inquiry serve significant roles in the design of or migration towards MSA, who are mainly composed of project managers (6/13) and senior software architects (7/13). The interviewees have 3 to 18 years of professional experience related to software architecture, over half of them have more than five years' experience in architecture design. Two senior software architects were responsible for the technology foundation platform underlying the migrated projects, while the rest took charge of the migration projects.

Fig. 2 and Table I respectively show the locations and the domains of the organizations that the interviewees worked with. The thirteen organizations are located in five majors cities in China and cover a wide variety of business domains. Three of our interviewees worked for Internet enterprises, two for bank and finance institutions, and one from an e-commerce company. Besides, three interviewees worked for the companies that provide software or cloud computing capability as a service, two for IT system integration enterprises, and two in telecommunication companies. These companies offer commercial software, information system solutions, or provide the technical capacity as s service for enterprise users.

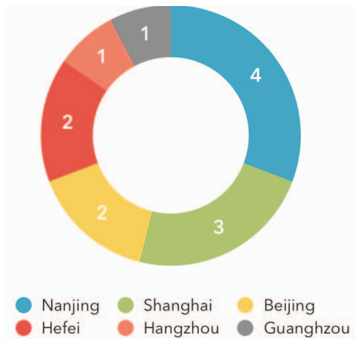

Fig. 2. Locations of participating organizations

TABLE I
BUSINESS DOMAINS OF PARTICIPATING ORGANIZATIONS

Domains	#
Internet	3
IT System Integration	2
Telecommunication	2
Bank and Finance	2
IT Service Provider	2
Cloud Provider	1
E-commerce	1

In order to gain an in-depth understanding of the state-of-the-practice of MSA, we took both completed and ongoing microservices projects into analysis.

IV. Findings

This study aims to reflect the state-of-the-practice of microservices in industry and bridge the gap between the visions in community and the reality in practice. During the interviews, the interviewees were enquired about their practical experiences (RQ1), benefits and pains perceived (RQ2) with microservices in terms of the nine characteristics claimed by Fowler and Lewis [2]. Unfortunately, we did not obtain any informative evidence related to the characteristic–"products not projects", which was not recognized as a 'pain' to any interviewee. Accordingly, this section reports the more evidential findings on the rest eight characteristics. For a clear presentation, the findings related to each characteristic of microservices are organized into two parts: one is the practice that the practitioners may benefit from, and the other is the possible pain brought by the characteristic.

A. Componentization via Services

1) Practice 1: **Independence by separation**

Microservices are defined to be out-of-process components who communicate with a mechanism such as a web service request, or remote procedure call [2]. Overall, the underlying motivation as well as the benefit of componentization are the

TABLE II
BENEFITS RELATED TO COMPONENTIZATION VIA SERVICE.

Practices	#
Independent upgrade	8
Independent scale (up/down)	4
Independent development	4
Independent testing	3
Independent deployment	2

independence or the autonomy of microservices, which has been realized by all the interviewees. Table II shows specific services are upgraded independently with MSA in a more efficient way (8/13), which is different from the complex manipulations on the monolith as a whole. Similarly, services are allowed to be scaled up and down easily due to their separation from each other, as mentioned by 31% interviewees (4/13). Moreover, through partitioning the monolith into a set of fine-grained components as services, it is possible independently develop (31%), test (23%) and deploy (15%) a service in one team, which would consequently shorten the lead time of products in practice.

*2) Pain 1: **Chaotic independence***

Nevertheless, participants do not always enjoy the benefits of componentization in MSA. Sometimes the independence of microservices is not properly realized for various reasons. As a result, some gains shown in Table II may turn negatively to become pains. For instance, inappropriate boundary among services may decrease the testability, especially when the microservices are over fine-grained. As the concern of about 46% interviewees (6/13), development or change of one feature may influence multiple microservices, which are distributedly developed and deployed. Moreover, microservices are released frequently and one microservice may have multiple versions simultaneously, which increases the complexity of joint debugging among different versions. In addition, the developers may face the requests of joint debugging from different developers at the same time. Some remote microservices are not always connectable in this process, which exacerbates the pain of testing.

"All these problems may seriously influence the speed of debugging and the efficiency of developments..."

B. Organized around Business Capabilities

*1) Practice 2: **Organizational transformation***

According to Conway's law [17], the technology architecture and the organizational structure interrelate to each other. As suggested by Fowler and Lewis, teams of an organization should be separated with reference to the business capability of microservices, and one (microservices-based) product or component is developed and operated by a dedicated team. This characteristic is confirmed by 77% (10/13) interviewees in their practice, and half of them mentioned that microservices could drive the partition of teams, which reduces the dependency and the communication among teams, and consequently lowers the cost of management. In opposite, the team structure may become a factor influencing the decision on the decomposition of microservice candidates even the technology architecture. Some common practices of transformation adopted in the interviewees' organizations are listed in Table III.

TABLE III
PRACTICES RELATED TO ORGANIZATIONAL TRANSFORMATION

Practices	#
Separation around business capability of microservices	7
Development and operation in one team	4
Dynamic evolution of teams	4
Increase of developers	2

It is observed that more than 50% interviewees (7/13) take team structure into account when decomposing services of legacy systems or designing new microservices.

"The ideal is that the development of one microservice could be assigned to 1 to 2 developers. Needing 3 to 5 developers to some extent indicates that the microservice can be further partitioned... "

Three organizations often allocate both developers and operators for each team in MSA. However, this vision can not be achieved in other organizations because of the shortage of staff. It is worth mentioning that two organizations experienced the increase of developers during transformation. Parallel development of microservices between different teams happens in all the interviewees' organizations, while one team may be responsible for developing multiple microservices. In addition, teams could vary dynamically along with the evolution of the microservices in three organizations. Once the development workload decreased, some team members may join in other teams.

*2) Pain 2: **Unguided organizational transformation***

Although drawing support from some strategies, organizations in this study still achieve the transformation mainly based on experience. As predicted by 23% participants (3/13), practitioners may suffer from the inappropriate organization transformation along with the migration to MSA. The mismatching between the organization structure and the architecture may cause the low efficiency of internal development or the massive communication among teams. Because it is difficult to ensure the 100% independent decomposition of microservices, to different extent interaction may exist due to the coupling of microservices, especially when it needs to do the debugging among relevant ones. Under this circumstance, it is doubtful that whether the microservices could improve the efficiency of internal development. One participant explained his idea and requirement about this problem as follows:

"We are aware that the organization should catch up with the change of architecture accordingly, but we lack the guide on how to do it. Most importantly, we need the approval of the decision makers..."

C. Smart endpoints and dumb pipes

*1) Practice 3: **Choosing communication protocol***

Compared with the Enterprise Service Bus (ESB) in SOA, Smart endpoints and dump pipes are an alternative communication approach favored by the microservice community,

TABLE IV
SELECTION OF THE COMMUNICATION PROTOCOL IN MSA

Concerns	#	Practices	#
Communication protocols and Frameworks	11	HTTP Restful API (Spring Cloud)	6
		RPC (Apache Dubbo)	3
		Lightweight messaging (Rabbit MQ)	2

the communication protocol of which commonly uses HTTP-based RESTful API and lightweight messaging. Our interviews find that these two protocols still form the mainstream in industry and have been applied by some famous microservice frameworks. For example, `Spring Cloud` based on the RESTful API is the most popular frameworks adopted by 46% (6/13) among our interviewees (as shown in Table IV). Surprisingly, although Fowler and Lewis [2] consider that "A naive conversion from in-memory method calls to RPC leads to chatty communications which don't perform well.", there are still three interviewees using the `Apache Dubbo` framework whose communication protocol is based on RPC.

TABLE V
FACTORS IMPACTING FRAMEWORK SELECTION

Factors	#
Maturity of the ecosystem	8
Community	8
Document	6
Previous experience	6
Open-source	3
Users' technology stack	2
Cost	1

We questioned the interviewees their reasons for why a framework or a technology was selected. The answers are displayed in Table V. Four main factors concerned by practitioners are the maturity of the technology's ecosystem (8/13), the activity of the community (8/13), the supporting documents (6/13), and technology stacks with previous experience (6/13). A technology's ecosystem, including the components or the integrations it supports, may predict whether it will develop well in the future. The activity of the supporting community and the richness of the documentation may determine that if the technology is easy to learn. Besides, open-source tools were considered by about 23% interviewees, with which they could easily carry out some secondary developments according to their requirements. Only one interviewee mentioned they cared about the cost of a tool. Two interviewees in the telecommunication domain made the decision according to the end-users' technology stack.

The frameworks like `Spring Cloud` and `Apache Dubbo` are widely adopted by te interviewees, because they provide multiple components necessary to develop MSA-based systems, e.g., service discovery, API gateway, load balancer. The amount of people using `Spring Cloud` is twice as much as the one using `Apache Dubbo` because of its open-source characteristic.

2) Pain 3: ***Complexity of API management***

One serious pain related to the communications among microservices is the complexity of the implementation and management of API, as mentioned by 31% interviewees (4/13). The APIs provided in MSA are required to conform to the contact, otherwise, there would be problems in the interaction of services. One interviewee mentioned that they encountered the problems of implementing the repetitive interface, which was not addressed. According to the description of other two interviewees, different teams may have the inconsistent understanding on the APIs, and the problems may not be identified as soon as possible due to the independent releases of service. Two interviewees responded that they had not found any effective way to ensure the consistent understanding of APIs. Complex internal regulations were adopted in two participating companies, where they normalized the API development through some pre-defined rules, e.g., naming rules, and then some basic verification was manually conducted to ensure that the developers implement the API by following those rules defined.

D. Decentralized Governance

1) Practice 4: ***Controlling technology diversity***

Different from the centralized governance allowing one single technology platform, microservices support a mix of multiple languages, development frameworks and data-storage technologies, i.e. technology diversity. This was justified by the utilization of multiple language stacks and frameworks at the same time (as shown in Table VI). Among the participants, 85% used more than one language stacks in their MSA-oriented systems, among which Java is the mainstream serving 67% interviewees.

TABLE VI
INDUSTRIAL PRACTICES OF DECENTRALIZED GOVERNANCE IN MSA

Concerns	#	Practices	#
Language stack	13	One (Java, C, C++)	3
		Two (Java, C, C++, Go, Python, JavaScript, etc.)	7
		More than two (Java, C, C++, Go, Python, JavaScript, etc.)	3

Although MSA supports different technologies, practitioners do not encourage much various diversity when selecting the suitable technology in this phase, because too many technologies may arouse the problems, e.g., versioning. Therefore they usually make the decision of technology selection according to the considerations displayed in Table V.

2) Pain 4: ***Excessive technology diversity***

As mentioned above, MSA is able to accomodate many technology stacks, which is not limited within the ones listed in Table VI. The diversity of technology stacks also raises the requirements on developers' ability and increases the complexity of the learning and construction.

It is worth mentioning that 23% interviewees (3/13) suggested that the high priority of technology architecture should be recognized. In other words, the design and the implementation of technology architecture can be done as early as possible, for example, at the beginning of considering migration towards MSA. Because learning new technologies and setting up technical frameworks may cost a longer time than expected. One interviewee told us a relevant lesson they learned by saying:

"We thought it was easy, so we unhurriedly selected and constructed the technology stacks only after the microservices having been decomposed from the legacy system. As a result, no product was delivered in the following two months, because all the people were busy in coping with the problems of excessive technology stacks..."

E. Decentralized Data Management

1) Practice 5: ***Compromise with database decomposition***

Although MSA advocates that each microservice manages its own database, we surprisingly identified that only three interviewees decomposed the database when migrating towards MSA. Accordingly, they experienced the pain of distributed transaction brought by MSA. To relieve this pain, databases were vertically decomposed according to the coarse-grained business processes (2/13). A compromising strategy was also adopted by one interviewee's team, in which they were separated into partial data tables owned by different microservices, and kept the rest as the shared component. Other interviewees (6/13) did not consider decomposing the database due to the following reasons:

- *Memory database in use*. One interviewee from the telecommunication domain used the memory database rather than the traditional type in their legacy system. Data is inherently tied to the relevant microservices, thus there is no need to decompose the database.
- *Databases partitioned in legacy systems*. Three interviewees expressed that they have implemented the snapshot of database and separated data tables into different databases using technologies like Binlog. They thought this design is basically suitable for their requirements in the MSA-oriented systems and it is unworthy of further decomposing the database into fine-grained ones.
- *Benefit analysis of decomposition*. As mentioned by two interviewees, although they were aware that decomposing the database contributes to decoupling the microservices, the problems (pains) after decomposition, such as the complexity of addressing data consistency, are prohibitive. In consequence, practitioners prefer to avoid the possible troubles and persist the shared database. Some traditional tactics are mentioned in their practices, for example, using caching to improve the performance, and using locks or the data storage scheme `ETCD` to ensure the data consistency.

2) Pain 5: ***Data inconsistence***

After migrating to MSA with multiple independent databases, practitioners are usually pained with the complexity of addressing the distributed transactions while ensuring the data consistency. Therefore, decomposing the database was seriously considered and actually realized into practice by few interviewees (2/13). One of them gave the following description about the pain of addressing data consistency:

"It troubles us for a long time. We need to consider different scenarios because you know different scenarios have different designs..."

MSA perplexes the inconsistency problems due to its distributed nature. Although the monolith is not free from this problem, it could be addressed easily by updating a bunch of things together in a single transaction. Nevertheless, it becomes complex to handle this issue in different scenarios of MSA because of the decentralized data management. As mentioned by one interviewee, most Internet companies prefer the 'eventual consistency' of the distributed data to the strong consistency, especially when their requirements on performance or availability are higher than the data consistency (CAP). Although some methods are available for addressing the eventual consistency, e.g., messaging-based or event-based, the design and implementation are very complicated. Once the eventual consistency is taken, with the risk of some inconsistencies resulted from errors of a third party, to a great extent human intervention is required to get the final result manually and then make the data consistent rather doing the rollback of data directly. The real process of addressing the data consistency is far more complicated than the description, which causes sufferings to the practitioners after the migration to MSA.

F. Infrastructure Automation

1) Practice 6: ***CI & CD***

Fowler and Lewis argue that many teams building microservices possess rich experience of Continuous Delivery (CD) and Continuous Integration (CI). By applying infrastructure automation technology, the deployment of microservices would be as simple as that of monolithic systems. To identify the practices of the infrastructure automation in the interviewees' organizations, we questioned them the automated tools being utilized in different phases of continuous delivery, including build & configure, test, and deploy, etc.

Fig. 3 displays various types of automation technology adopted by the interviewees in the phase of developing microservices (Build & Config). `Jenkins` is the most popular CI tool, which was used by about 62% participants (8/13). It is worth mentioning that one interviewee applied the new service mesh technology to facilitate the upgrade and the version management of microservices.

2) Pain 6: ***Inadequate automation***

As for the construction of high-level automated infrastructure, 38% participants (5/13) complained that many existing infrastructure-related technical frameworks are unsatisfying to meet their requirements. On one hand, foreign technologies and tools spent them much effort and cost on setting one by one up. The relevant communities are not active enough, and also the incomplete documents may make this pain more serious. On the other hand, as an important part of infrastructure automation, container and its management bring difficulties to 23% interviewee in their practices, for example, container management, and automatic scaling in response to the peak flow.

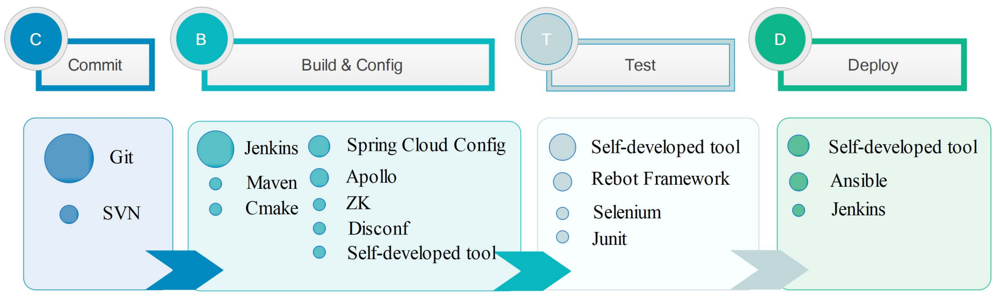

Fig. 3. Tool stacks used for infrastructure automation.

G. *Design for failure*

1) Practice 7: ***Monitoring and logging***

The "Design for failure" characteristic emphasizes and relies on sophisticated monitoring and logging setups for microservices. We enquired the interviewees about their practices of monitoring and logging, and identified some experiences accumulated in industry.

a) Monitoring metrics: Various metrics have to be checked and visualized in the real-time monitoring of MSA-oriented applications. Table VII displayed the major metrics that interviewees focused on. Among 13 interviewees, ten mentioned that they paid attention to the architectural elements of the systems using MSA, like CPU, memory, and call chain graph of services; eleven mentioned that they monitored some business-relevant metrics, including transaction volume, transaction success rate, and failure rate.

TABLE VII
MONITORING METRICS.

Type	Metrics	#
Architectural elements	Memory	5
	Latency	4
	CPU	3
	Infrastructure Topology	3
	Status of Services	2
	Query Per Second (QPS)	2
	Load	2
	Security	1
Business relevant metrics	Transaction Volume	3
	Transaction Success Rate	2
	Requests Per Second	2
	Errors	2
	Daily Activate Units	1
	Customer Retention Rate	1

b) Monitoring strategies: Common monitoring strategies adopted by our interviewees are log analysis (8/13), automatic alarming, call chain tracking and dashboard. Three interviewees (23%) said they used ELK stack (i.e. `Elasticsearch`, `Logstash` and `Kibana`) for the monitoring, search, and visualization of logs; nine (69%) indicated that the automatic alarming strategy was used in microservices' monitoring, while the system's setting threshold depends on previous experience; seven paid special attention to troubleshooting and location of call chains of microservices, through using distributed tracing systems like `Zipkin` and open source APM tools like `Pinpoint`; and six (46%) replied to us that visual dashboards were relied on to monitor microservices, such as `Grafana`, `Kanban` and self-developed tools.

2) Pain 7: ***Unsatisfying monitoring and logging***

The distributed nature of microservices brings several challenges of monitoring MSA systems. All interviewees in our study expressed the pains caused by distributed microservices, which can be described into the following four aspects.

a) Insufficiency of automated operation: When talking about the automatic monitoring of microservices, four interviewees (31%) complained to us that the current automation was not satisfying. Some operation work still depends on checking the logging manually, which is time-consuming and inefficient.

b) Difficulty in troubleshooting: A microservices-based system may include hundreds of instances of microservices, so it is usually difficult for practitioners to locate problems among large amount of distributed microservices. Three interviewees (23%) reported that they still relied on traditional troubleshooting methods, such as logging and distributed tracking.

c) Difficulty in threshold setting: Setting appropriate alarming thresholds in MSA-oriented systems is also painful for many practitioners, which relies on the previous experiences and lacks methodological guidance. The improper threshold settings are likely to be accompanied by subsequent misjudgments and omissions of anomalies or even faults. Moreover, the threshold settings need continuous artificial adjustments.

d) No well-rounded monitoring system: Microservices put forward new requirements for systems' monitoring and operation. Four interviewees (31%) have expressed that it is troublesome to switch between different monitoring platforms, which are only able to monitor partial metrics. They desired a monitoring platform to provide the monitoring and the visualization of microservices from multiple dimensions, including the hardware, middleware, and software, to form an integrated monitoring scheme.

H. *Evolutionary Design*

1) Practice 8: ***Stepwise evolution***

In common with the evolution characteristics of microservices described by Fowler and Lewis, our interviewees confirmed that no monolithic system could suddenly migrate to MSA without transition and the separation of microservices from the monolith should be done step by step. Apart from this common practice, we also identified the major concerns of practitioners when designing the microservices-oriented systems: *understanding the legacy systems*, *determining the boundary of new microservices*. The practices corresponding to these concerns are shown in Table VIII.

TABLE VIII
CONCERNS AND PRACTICES WHEN DESIGNING MSA

Concerns	#	Practices	#
Legacy system	5	People	3
		Designing documentations	3
		Source code	5
Boundary of microservices	8	Vertical decomposition	5
		Horizontal decomposition	4
		Domain-driven design	3

Understanding of legacy systems: When microservices-based systems are migrated from legacy systems rather than developed from a green-field design, it is necessary to understand the business logics and the architecture of legacy systems to gain the knowledge for microservices. From the interviews, we observed that experienced developers or architects, as well as designing documentations play an important role in completing this work. However, five interviewees mentioned that they experienced many difficulties to achieve this, because the legacy systems in their companies had been working for a long life with the absence of necessary design documents. For example, one interviewee from a banking organization described one of their legacy systems as follows:

"The system has been more than 30 years old. We have nothing but the puzzling source code. You can not imagine how complex it is! It is hard for us to understand that monster, not to mention the decomposition of it..."

Under the circumstance like this, practitioners have no choice but relying on the costly manual code reading and analysis of the dependency among components. Moreover, the technology stacks of these systems are often out-of-date so that no reverse engineering tool is available to improve the productivity of the time-consuming job.

Boundary of microservices: Identifying the boundary of microservices to determine the appropriate microservice candidates is another concern addressed by the majority of practitioners (8/13) in the design phase. The overall principle of MSA-oriented decomposition is to achieve "high cohesion and low coupling". Following this high level principle, three types of decomposition methods can be adopted for boundary identification of microservices in practice:

- *Vertical Decomposition* is the most common strategies in industry (5/13) that decomposes services vertically according to the business capabilities. Businesses with the similar domain tend to be aligned to the same microservice, which reduces the coupling among them. While different business capabilities often correspond to different high-level domain objects, which are easy for practitioners to identify, e.g., the Order management and the Product management in the online store system.
- *Horizontal Decomposition* is another strategy in which business processes of a system are partitioned into different layers. This method extracts some core and common components, including the non-functional ones, to be fundamental microservice candidates and improves the reusability of them. In practice, this strategy is usually combined with the vertical method to decompose the legacy systems from different dimensions. Among our interviewees, 31% (4/13) mentioned that they applied both the strategies in their identification of microservice candidates.
- *Domain-Driven Design* (DDD) from Evans [18] is the third way by which three interviewees identified the boundary of microservices. DDD offers many methods to help practitioners analyze the business process of legacy systems, construct the domain models and the bounded context, and finally determine the boundary of microservices. Popular methods of DDD are *Brain Storming*, *Event Storming*, and *Four Color*. One interviewee applied the first two methods in the decomposition and felt that the modeling process of *Event Storming* is more suitable for them. Because time consumed and the results of *Brain Storming* method usually depend on the degree of understanding of the legacy system. In contrast, *Event Storming* is more controllable, in which a structured process is defined to guide the practitioners to obtain the aggregations. With *Four Color* method, according to the business processes introduced by relevant people or collected through some operation tools, roles and roles' process are analyzed and labeled by different colors to deliver sub-domains related to microservice candidates.

TABLE IX
FACTORS CONSIDERED IN THE MICROSERVICES-ORIENTED DECOMPOSITION.

Factors	#
The previous organizational structure	3
The frequency of modification	2
The need for scalability	2
The need for reusability	2
The influence on extensibility	2
The existing database	1

The decision on which service should be separated from the monolith is made by experienced architects in most cases. The common practice is that architects or decision makers prioritize the candidate microservices, according to one or more factors they may most concerned in Table IX, and then dissect the monster step by step. In the listed factors, the previous organizational structure is considered by three interviewees when conducting the decomposition, for the structure of their organizations were originally setup based on different business modules.

*2) Pain 8: **Missing of an effective decomposition guide***

During the design of MSA, the missing of an effective decomposition guide seems to be the most painful aspect for practitioners. In practice, it is the architect's experience that plays an important role in the process of decomposition. As mentioned by more than half of the interviewees (8/13), they did not have a systematic guide on decomposition and the decision was usually made based on experience. It does not represent that they all agree to be guided by experience, because some interviewees have the following consideration:

"Experience can tell us how to do it, but it can not tell us why we should do it..."

Although many theoretical strategies attempt to be adopted in industry, they are not so effective and can not meet the practical requirements of practitioners. For example, one interviewee complained about the problems of applying DDD in decomposition as follows:

"DDD is a relatively abstract thing to us, without a specific process. Since different people had different consensus of views about it, we spent a lot of time and effort to grope its application and unify the understanding."

V. Discussion

This study identified the common practices and pains of microservices in industry around the nine best known characteristics claimed by Fowler and Lewis [2]. The emerging findings about the pains (P1 denotes Pain 1) may inspire some research directions (RD1 denotes Research Direction 1) which can be further researched.

1) P2-RD1: Empirical research on organizational transformation

Conway's law [17] indicates that the (product) architecture is influenced by the organizational structure. The inverse law was also justified by five interviewees, who mentioned that microservices had impacts on the decomposition of their teams. Three interviewees thought that the mismatch between the organization structure and the (product) architecture may slow down internal development and/or lead to extra communication overheads among teams. However, this argument lacks solid support of evidence, no matter quantitative or qualitative. Empirical research on organizational transformation is advocated to help the understanding of the influences of different kinds of transformation, then identify their application contexts.

2) P1,P8-RD2: MSA-oriented decomposition & migration

Identifying the boundary of microservices and determining the appropriate granularity of microservice is a critical concern of 62% interviewees. Microservices at inappropriate granularity levels or mistakenly separated may make the pain (P1–chaotic independence) even more serious and consequently result in many consecutive problems, e.g., the extra complexity of joint debugging and the excessive communication among teams.

However, as reported by our interviewees, participants lack an effective guide on service decomposition (P8) and the decisions are commonly made by some experienced architects. Consequently, researches are encouraged to focus on such an MSA-oriented decomposition or migration method with some specific concerns of practitioners into consideration, e.g., the prioritization of different factors in Table IX, which is supposed to deliver different decomposition or migration strategies, including the boundary of microservices, benefit analysis and risk assessment of each decomposition strategy in specific case.

VI. Threats to Validity

An extremely cautious attitude was taken to mitigate threats when we designed and conducted this study. Specifically, the main threats to validity and corresponding mitigation strategies were considered while characterizing the target population, designing and wording the questionnaire, interviewing with participants, and the phases of data analysis and synthesis.

Construct Validity: The most recurrent threats in questionnaire-based survey are regarded as the phrasing adopted to define and refine the statements and questions in the questionnaire. To mitigate these threats, we firstly defined and refined the questionnaire continuously with final 8 major overhauls and several minor alterations. Secondly, a pundit who is experienced in research of unstructured interviews was invited to check the validity and integrity of the questionnaire. Thirdly, we conducted a pilot study with internal researchers and external random selected participants and modified the questionnaire extensively.

Conclusion Validity: To mitigate this threat, we exploited theme coding and inter-rater reliability assessment to limit observer bias and interpretation bias. To be specific, 1) a qualitative data analysis software–NVivo, was employed to help theme coding; 2) we also cross-analyzed the answers of different questions from every interviewer, and then compared every extracted data to find disagreements; 3) every disagreement found in data analysis and synthesis was discussed until resolved.

External Validity: Representativeness of the data may be a potential issue of our study. To migrate this threat, all participants of our survey were obtained from sampling. Specifically, we invited participants who work for enterprises of seven domains and five different cities based on careful selection of hundreds of interviewee candidates obtained from three famous industrial conferences. Varying levels of experiences of our interviewees ensures the generalizability of our research.

VII. Related Work

With the increasing interest on microservices, a few systematic mapping studies related to MSA have been recently reported. Pahl et al. [7] conducted a systematic mapping study on the motivation, type, techniques and challenges of microservices. Focusing on taxonomically classifying and comparing their 21 identified studies, they proposed a characterization framework. Alshuqayran et al. [4] reported another systematic mapping study on microservices. Aimed at presenting an overview of the existing challenges, architectural diagrams/views and QAs, 33 relevant studies were identified and analyzed in their review. Similar to the aforementioned reviews, Francesco et al. [8] performed a systematic mapping study on the trends, focus, and potential directions of MSA most recently. All these secondary reviews were based on the academic studies and did not directly address the migration to MSA. By contrast, our work involved the industrial experiences from practitioners and specifically focus on the practices and challenges when and after adopting MSA.

There are several studies which directly investigate the migration towards MSA. Chen [13] observed benefits and new challenges associated with MSA. He shared the practical strategies that can be employed to address these new challenges, discussed situations for which MSA may not be a good choice, and outlined areas that require further research. However, all the experiences and observations stem from one company and only represent one data point. Di Francesco et al. [10] performed an industrial survey on migration practices towards the adoption of MSA. The investigated the migration activities and challenges in three phases. Taibi et al. [9] conducted a similar questionnaire survey on migration towards MSA with different focus, e.g., the motivations behind the migration to MSA, and the benefits that MSA provides. Our work differs from these two studies in two ways: we focused on the gaps between the visions and the industrial practices from the characteristic dimensions widely-accepted both in academia and industry; we also paid attention to the pains and the potential research directions of all those dimensions, which were not discussed in much detail in their study. Soldani et al. [19] identified the gains and pains through analyzing the industrial grey literature. By contrast, we performed in-depth interviews with industrial software professionals and investigated the industrial practices, new pains and potential solutions of MSA.

VIII. Conclusion

Microservices is the latest trend of architectural style, which has attracted growing interest of industry and academics. Many initiatives have contributed to understanding various aspects of microservices, e.g., benefits and challenges. However, the knowledge construction in this domain is still on the road. Gaps between the visions and the reality of microservices need to be empirically investigated. This study aimed to gain an in-depth understanding of the gaps between the community-wide accepted visions and the real practices of microservices, including the practical experiences with specific ideal characteristics of microservices, the reasons behind the gap, and the pains that should be mitigated. We carried out an empirical study by applying the semi-structured interviews to collect data from thirteen experienced participants in a variety of domains of software industry. Based on the data collected, we characterized the gaps in terms of nine best known characteristics of microservices. Furthermore, the trade-offs between benefits and sufferings of microservices in these nine dimensions were analyzed. The findings of industrial experiences related to these grand visions of microservices will benefit the practitioners for their exercises in adopting and implementing microservices. Besides, the pains associated to some aspects of microservices, e.g., organizational transformation, decomposition, monitoring, and logging, may also inspire researchers with further research directions.

This study is the initial stage of our research program. In the next stage, we plan to involve more interviews with practitioners to capture more detailed sufferings and practical solutions of MSA in industry. The research outcome may be complemented with the multivocal literature review [20] or a questionnaire based survey in a larger population to validate and generalize the important findings.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No.61572251).

References

[1] S. Newman, *Building microservices*, 2nd ed. Sebastopol, California: O'Reilly Media, 27 February 2015.

[2] M. Fowler and J. Lewis, "Microservices - A definition of this new architectural term," 10 June 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html

[3] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 35, no. 3, pp. 24–35, 2018.

[4] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE Press, November 2016, pp. 44–51.

[5] A. Furda, C. Fidge, O. Zimmermann, W. Kelly, and A. Barros, "Migrating enterprise legacy source code to microservices: On multi-tenancy, statefulness and data consistency," *IEEE Software*, vol. PP, no. 99, pp. 1–1, 2018.

[6] S. Hassan and R. Bahsoon, "Microservices and their design trade-offs: A self-adaptive roadmap," in *Proceedings of the 2016 IEEE International Conference on Services Computing (SCC)*. IEEE Press, June 2016, pp. 813–818.

[7] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study," in *Proceedings of the 6th International Conference on Cloud Computing and Services Science*. ACM, April 2016, pp. 137–146.

[8] P. D. Francesco, I. Malavolta, and P. Lago, "Research on architecting microservices: Trends, focus, and potential for industrial adoption," in *Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE Press, April 2017, pp. 21–30.

[9] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation," *IEEE Cloud Computing*, no. 5, pp. 22–32, 2017.

[10] P. Di Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: an industrial survey," in *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2018, pp. 29–2909.

[11] S. Kvale, *Doing interviews*. Sage, 2008.

[12] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on software engineering*, no. 4, pp. 557–572, 1999.

[13] L. Chen, "Microservices: Architecting for continuous delivery and devops," in *IEEE International Conference on Software Architecture (ICSA)*, 2018.

[14] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[15] E. Welsh, "Dealing with data: Using nvivo in the qualitative data analysis process," in *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, vol. 3, no. 2, 2002.

[16] M. Q. Patton, *Qualitative evaluation and research methods*. SAGE Publications, inc, 1990.

[17] M. E. Conway, "How do committees invent," *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.

[18] E. Evans, *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.

[19] J. Soldani, D. A. Tamburri, and W.-J. V. D. Heuvel, "The pains and gains of microservices: A systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215 – 232, 2018.

[20] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering: Complementing systematic literature reviews with grey literature," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016)*. Limerick, Ireland: ACM Press, Jun. 2016, pp. 26:1–26:6.