Empirical Software Engineering (2025) 30:127
https://doi.org/10.1007/s10664-025-10684-4

Balancing technology heterogeneity in microservice architectures

Georg-Daniel Schwarz[1] · Philip Heltweg[1] · Dirk Riehle[1]

Accepted: 3 June 2025
© The Author(s) 2025

Abstract
Microservices are a popular architectural style that allows systems to be built from a potentially large number of microservices, all of which can be developed independently and by their own teams. As a resulting benefit, development teams can choose the technologies optimal for their microservices, leading to a diversity of different programming languages, frameworks, and further technology in use. However, this heterogeneity presents challenges as it prevents code reuse and complicates moving individuals between microservices due to knowledge hurdles. We performed 15 expert interviews in a qualitative survey to build a theory on how technological heterogeneity can be balanced in microservice architectures to reach a context-dependent compromise between its benefits and drawbacks. We contribute by (1) gathering empirical data from industry professionals on a research topic that has been acknowledged but has only seen limited exploration so far, (2) developing a comprehensive theory of technology heterogeneity as a major integration challenge in microservice-based projects, (3) proposing a framework to overcome the challenge of balancing technological heterogeneity in microservice architectures, (4) optimizing the theory's presentation for practical use in industry by using the well-known pattern format, and (5) generating research hypotheses to guide and inspire future investigations into this phenomenon.

Keywords Microservices · Technologies · Technological heterogeneity · Integration · Governance · Best practices

Communicated by: Xin Peng.

✉ Georg-Daniel Schwarz
georg.schwarz@fau.de

Philip Heltweg
philip@heltweg.org

Dirk Riehle
dirk@riehle.org

[1] Computer Science Department, Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany

1 Introduction

Microservices have gained widespread adoption as an architectural style for building robust, scalable, and suitable software systems for cloud environments (Jamshidi et al. 2018). Instead of building one big monolithic application, the system is split into cohesive and loosely coupled units, each with distinct responsibilities, called microservices. Depending on the project's maturity, microservices can be deployed independently without requiring changes to other system components, and their development lifecycles may be completely independent. The loose coupling from a domain perspective and the loose organizational coupling of microservices teams enable parallel development of large software projects by multiple teams (Newman 2021).

Microservice-based systems are distributed systems that present unique challenges. For example, unlike monolithic applications, microservices cannot rely on in-process communication but instead must integrate over the network. As a result, complexity shifts from the application layer to the network and integration layer. Transitioning from a monolith to microservices makes integration a predominant and more explicit challenge (Baškarada et al. 2018).

Contrary to monolithic applications that are packaged as one big application, microservices operate independently and do not share a process or a runtime, such as a JVM. By lifting the restriction of being packaged together, each microservice has the potential to utilize a completely different technology stack (Krylovskiy et al. 2015). Differences in technologies may cover using different programming languages, frameworks, libraries, runtimes, communication protocols, etc. The architecture itself enables this diversity of technology but doesn't imply a necessity for it to emerge. Relaxing centralized governance of standards and technology platforms enables technological heterogeneity (Krylovskiy et al. 2015). Projects may adopt different technologies for each microservice (complete heterogeneity), a single technology across all microservices (complete homogeneity), or something in between those two extremes. Thus, we speak of the degree of technological heterogeneity, which is specific to each project.

This technological freedom presents opportunities like the selection of the best tool for a given situation. The system can gradually adopt new technologies by re-implementing one microservice at a time (Krylovskiy et al. 2015). This approach also fosters a culture of innovation and experimentation (Bogner and Zimmermann 2016).

However, technological heterogeneity presents a significant challenge to expanding projects as the number of technologies used grows (Di Francesco et al. 2018). Compared to technological uniformity as the antagonist, the effects of building and sharing knowledge among teams and utilizing in-house libraries and operational efforts cannot be effectively adopted due to the diversity of technology. Additionally, moving individuals between teams becomes more challenging due to the increased ramp-up time required for learning the technologies of another microservice (Chen 2018).

The degree of technological heterogeneity in a system is a trade-off between its benefits and drawbacks:

> *"[Technology heterogeneity] behaves like many things. The value curve is [...] a parabola, logically speaking. It has a sweet spot. Too little is not good; too much isn't good either."* - Interview G4, translated from German

Thus, we provide readers with a toolkit to converge toward a context-dependent compromise by answering the following research question:

RQ How to balance the benefits and drawbacks of technical heterogeneity in microservice architectures?

To answer the research question, we built a theory on balancing technological heterogeneity by performing a qualitative survey with a total of 15 conducted expert interviews (Section 3.1) and applied thematic analysis (Section 3.2) for data analysis.

A theory consists of abstract knowledge aimed at explaining a phenomenon or predicting outcomes and is primarily a set of (typically interrelated) hypotheses rooted in data. This article presents such a theory in the format of a multi-tier influence research model and an actionable and practitioner-relevant set of 13 best practices.

The primary contributions of this article are as follows:

– We present a comprehensive theory of technological heterogeneity as a major challenge in integrating microservices.
– We present a collection of best practices to balance technological heterogeneity. For each best practice, we describe the problem it addresses and in which context it is applicable, followed by a reusable solution. By using this actionable format, practitioners can use those patterns as a guide to overcome their project-specific challenges.
– Our article advances the field of microservice research by highlighting the organizational aspect of balancing technological heterogeneity that, to the best of our knowledge, has not been covered in such depth in academic literature.
– We generate hypotheses based on our insights to guide and inspire future research.

We first discuss relevant related literature in Section 2. Section 3 presents our research methodology and data analysis methods, followed by the results of the study in Section 4. The implications of our findings and their potential applications are outlined in Section 5. Section 6 outlines the study's limitations. Finally, we summarize our contributions and suggestions for future research directions in Section 7. Supplementary materials to browse the results are made available.[1]

2 Related Work

Governing heterogeneity in IT systems is a topic that has found attention in research. For example, Widjaja and Gregory (2012) propose design principles for heterogeneity principles in enterprise architecture (EA) management. They define ”IT heterogeneity as diversity of attributes of components in an EA“ and take infrastructure elements but also business processes into consideration. We extend their work in a more specific and narrow context. Our study focuses on technological heterogeneity, a subset of IT heterogeneity, and the context of microservice-based systems.

Few studies investigate the technological heterogeneity of microservice-based projects as a secondary subject. Di Francesco et al. (2018) cite one of their interviewees that unifor-

[1] Supplementary materials are available on Zenodo: https://doi.org/10.5281/zenodo.15632039

mity across their services is currently their most significant challenge. Chen (2018) points out the consequences of technological diversity as increased operational overhead. They mention their solution is putting technologies under governance by employing a review process. We build on their work by adding more depth to the challenge of technological heterogeneity and its implications, as well as detailing the governance process and techniques as potential alternatives to classical top-down review processes.

Rademacher et al. (2019, 2020) propose approaches to streamlining different technologies by using aspect-oriented modeling and meta-modeling in a microservice-based system. By modeling technologies within an architecture, the decision for and against introducing new technologies becomes explicit. We extend their work by including organizational processes and other measures to cope with technological heterogeneity. At the same time, our research approach differs. While their studies each propose a new technique and evaluate it in a case study, we use expert interviews to learn from practitioners about techniques that are already successfully used in practice.

Existing literature presents patterns and best practices in the context of microservice architectures from different viewpoints. Balalaie et al. (2018), for example, present migration patterns, Harms et al. (2017) present patterns related to front-ends. Weerasinghe and Perera (2022) present patterns for service decomposition, data management patterns, deployment, APIs, service discovery, and resilience. Márquez and Astudillo (2018) evaluate the application of communication, orchestration, deployment, and backend patterns in open-source projects. Taibi et al. (2018) present a pattern catalog with deployment patterns, data storage patterns, and patterns for orchestration and coordination.

These studies present valuable and actionable patterns by focusing on the architectural, technological, and operational aspects. We complement their work by shedding light on an organizational topic, the balancing of technological heterogeneity within microservice-based projects. In contrast to their work, who present practical experiences in pattern form based on the experiences of their authors or use systematic reviews, our study relies on first-hand empirical industry data by way of a broad interview study.

In summary, we present an in-depth investigation of how to balance technological heterogeneity when building and maintaining microservices-based systems. In addition to our work on technical challenges and beyond prior and related work, we also include organizational challenges as a key part of our theory.

3 Research Approach

Figure 1 gives an overview of the overall research approach we applied to answer the following research question:

RQ How to balance the benefits and drawbacks of technical heterogeneity in microservice architectures?

We built a theory on balancing technological heterogeneity in microservice-based projects by conducting a qualitative survey (Jansen et al. 2010) based on 15 expert interviews (Section 3.1). We applied thematic analysis for data analysis and synthesis (Section 3.2).

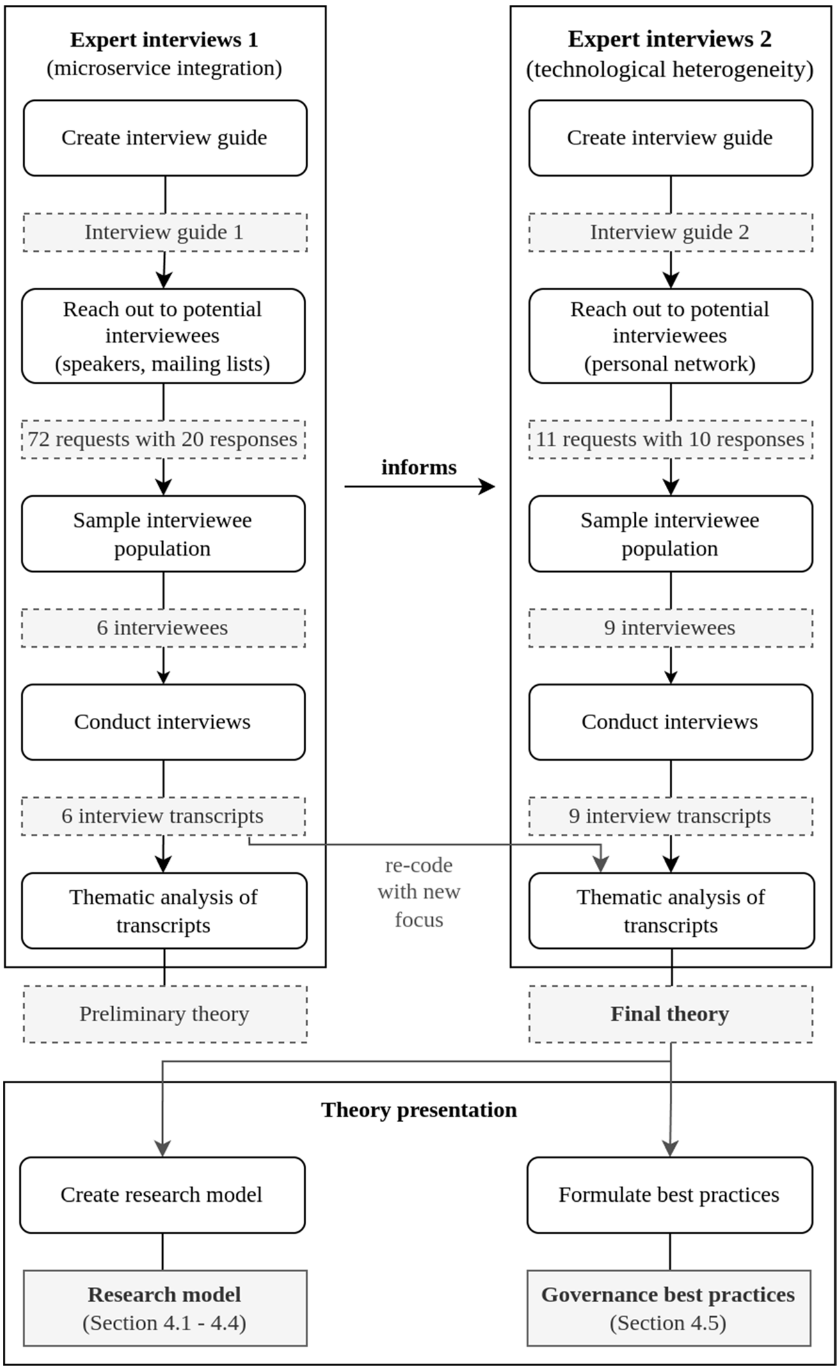
Expert interviews 1
(microservice integration)
Create interview guide
Interview guide 1
Reach out to potential interviewees (speakers, mailing lists)
72 requests with 20 responses
Sample interviewee population
6 interviewees
Conduct interviews
6 interview transcripts
Thematic analysis of transcripts
Preliminary theory
informs
re-code with new focus
Expert interviews 2
(technological heterogeneity)
Create interview guide
Interview guide 2
Reach out to potential interviewees (personal network)
11 requests with 10 responses
Sample interviewee population
9 interviewees
Conduct interviews
9 interview transcripts
Thematic analysis of transcripts
Final theory
Theory presentation
Create research model
Research model (Section 4.1 - 4.4)
Formulate best practices
Governance best practices (Section 4.5)

Fig. 1 Research approach

Section 4 presents the resulting theory as two artifacts. The first is targeted to the scientific community: a structured research model that describes about the relation between different influencing variables when dealing with technological heterogeneity. The second is targeted to the practitioner community: a list of governance best practices on how to balance the technological heterogeneity in industry projects.

3.1 Expert Interviews

We conducted two iterations of expert interviews (see Table 1). The first iteration was conducted in the context of a prior study (Schwarz et al. 2025) to explore the challenges of microservice integration and their solutions broadly. Among other topics, the six conducted interviews pointed towards balancing technological heterogeneity as a major integration challenge. We also found first solution approaches, such as introducing standardization, but in a broader context of simplifying coordination among microservice teams. Those results motivated us to deepen our understanding in a follow-up iteration of expert interviews that was narrowly focused on the topic of technological heterogeneity. We used the insights to design the research question and the interview guide for the second set of interviews. Further, we used the interview transcripts as the first iteration of this follow-up study.

In this second iteration, we conducted nine interviews to specifically answer the RQ on balancing technological heterogeneity in microservice-based projects. We re-coded the interview transcripts from the first iteration in relation to the updated research question to strengthen the evidence of the best practices on how to balance technological heterogeneity with additional insights.

For all 15 interviews, we followed the qualitative survey approach defined by Jansen et al. (2010).

3.1.1 Interviewee Sampling

To select suitable interviewees, we first created a sampling model (see Table 1; includes the sampled population). It contains fine-grained categories towards factors that might influence how microservice integration is facilitated: the interviewee's company, project, role, and experience.

As a quality measure, we asked an established expert in the field to provide feedback and confirm the chosen categories. We employed generic classifications for the expert's role since each company might define its specific roles with more fine-grained responsibilities that would be difficult to match against each other. On a higher level, we distinguish between in-house employees and consultants, as the latter tend to experience many different project contexts.

For the first set of interviews on microservice integration challenges in general, we utilized our group's network, the mailing list of the working group for microservices and DevOps by the German Informatics Society. We contacted 72 speakers at practitioner conferences like microxchg[2] or Microservice Summit.[3] We received 20 answers from willing interviewees, arranging them into our sampling model by filling out an online form.

[2] https://microxchg.io/2020/index.html

[3] https://microservices-summit.de/

Table 1 Interview sampling

Category / Feature		#	Interviews
Expert role	High-level consultant	(3)	G2, G5, T3
	Low-level consultant	(1)	G4
	Architect	(3)	G6, T1, T2
	Project manager	(2)	G3, T4
	Developer	(4)	G6, T5, T6, T7
	Operator / DevOps	(3)	G1, T8, T9
	Other	(1)	G4
Phase	Innovation	(1)	G3
	New software	(4)	G5, T3, T6, T9
	Rewrite	(1)	G1
	Evolution	(7)	G2, G6, T1, T2, T4, T5, T7
Teams	1 team	(1)	T5
	2-10 teams	(7)	G2, G3, G6, T2, T3, T4, T7
	10+ teams	(5)	G1, G5, T1, T6, T9
Services	1-10 services	(3)	G3, G6, T6
	11-50 services	(8)	G2, G5, T1, T2, T3, T4, T5, T7
	50+ services	(3)	G1, T8, T9
Deploy	Customer-managed	(4)	G1, G5, G6, T6
	In-house	(3)	G1, T3, T7
	Cloud	(8)	G2, G3, T1, T2, T4, T5, T8, T9

For the second set of interviews on technological heterogeneity in micro-service-based systems, we utilized our research group's network, personal contacts, and snowballing to reach out to practitioners. We contacted 11 practitioners and received ten answers. We base the significantly better response rate than the first set of interviews on relying more on personal connections and targeting specifically practitioners we knew were working on the microservice-based projects.

When selecting the interviewees for the study, we conducted polar sampling to capture the diversity of experiences with technological heterogeneity in the industry by covering each category within the sampling model adequately. According to Eisenhardt and Graebner (2007), using polar types of samples "leads to very clear pattern recognition of central constructs, relationships, and logic of the focal phenomenon".

Table 1 presents the sampled population arranged in the sampling model's major categories, showing our sample's diversity. Please note that we did not receive complete answers from G4 and T8 for the project-related questions. The main reason was that they wanted to elaborate on their experience across multiple project contexts rather than focusing on one specifically.

3.1.2 Interview Preparation

We followed the five phases presented by Kallio et al. (2016) to prepare for both sets of interviews, leading to two interview guides as artifacts.

Phase 1 - Identifying prerequisites: We first evaluated the appropriateness of semi-structured interviews according to our research questions. Semi-structured interviews allow us to study different organizational contexts and angles on the topic for a diverse perception and to discover topics especially relevant to practitioners.

Phase 2 - Previous knowledge: A preceding literature review study gave us a comprehensive understanding of the domain to prepare and conduct semi-structured interviews. We utilized our insights to construct the interview guides.

Phase 3 - Preliminary interview guide: We used the knowledge previously gained from the first set of interviews on general microservice integration for the second set of interviews on balancing technological heterogeneity in microservice-based architectures. We structured the interviews into multiple phases. Each phase consists of questions that allow the interview to steer toward our area of interest but are flexible and loose enough to allow open conversation. We led with open questions to get an unbiased response from the reviewer. Afterward, we followed up on interesting points the interviewees themselves or previous ones brought up. For example, the interview guide of the second iteration details the relation of technology heterogeneity to costs and innovation speed - two interesting features we condensed from the first set of interviews.

Phase 4 - Pilot testing: The interview guides were reviewed internally by members of our research group to avoid ambiguous or leading questions. We applied live field-testing by reviewing the interview guide after each interview, allowing for incremental improvements.

Phase 5 - Presenting the interview guide: Appendices A.2 and A.3 contain both interview guides.

We sent the interview guide with additional notes to our interviewees before the interview. Understanding the context and scope of the interview allowed them to prepare thematically and mentally. We included the following information:

– The context of our research.
– The process of an interview (time frame, the way we ask questions).
– The data assessment process (audio recording, interview transcription).
– The approval process: We send out each interview transcription to the interviewee to correct errors and misunderstandings. We use the interview transcription for further analysis only after approval by the interviewee.
– Data confidentiality and privacy (data pseudonymization for data analysis and anonymization in publications).

3.1.3 Data Collection

We found the interview guides especially useful to streamline our interviews. They supported us in sticking to the semi-structured frame and avoiding deviations from the topic of interest. In the first set of interviews, we used the guide as a checklist rather than sticking to it strictly, as the topic was broad. In the second set of interviews, we strictly followed the interview guide but still allowed for spontaneous follow-up questions where we saw fit.

After the interview, we transcribed the audio recording. The transcript was sent to the interviewee for review to detect misunderstandings and consider second thoughts on some of the insights they gave us. After their final approval, we added the interview transcripts as primary materials for analysis.

Myers and Newman (2007) present interview situations as a dramaturgical model. We followed their guidelines to prepare for an ”excellent performance“:

1. *Situating the researcher as actor:* Next to our initial letter to the interviewees, we devoted a section in the preamble phase of the interview to introduce us, our research, and the goal of the particular interviews. This step was especially helpful in familiarizing the interviewees with our research and setting the context for the interview.
2. *Minimize social dissonance:* To create a comfortable and natural environment for the interviewees, we included some small talk in the preamble phase of the interviews. We offered every interviewee to review the interview transcription to establish trust and allow for second thoughts after the interview as a quality measure.
3. *Represent various ”voices“:* We sampled for different roles in the organizations to avoid one predominant ”voice“ to emerge - see Section 3.1.1.
4. *Everyone is an interpreter:* We acknowledge in Section 6 that conducting the interviews and analyzing their transcripts have an interpretative factor.
5. *Use Mirroring in questions and answers:* We avoided imposing our language on the interviewees by mainly asking open questions, giving time for thinking, and elaborating on answers. We practiced mirroring by repeating phrases of previous answers to questions, moving from more general to specific topics. This technique aligns with the verbal and non-verbal probing techniques Kallio et al. (2016) recommend. Those probing techniques during the interviews enhanced the accuracy and clarity of both questions and responses, as well as supported us to uncover hidden information.
6. *Flexibility:* We applied semi-structured interviews with an interview guide, leaving details open for improvisation. The guide allowed us to follow up on topics the interviewee seemed most knowledgeable about, comfortable with, and confident in.
7. *Ethics of Interviewing:* We explained the confidential handling of obtained permission to use the interview data individually before, during, and after the interview: In the initial letter to interviewees, the preamble of the interview, and the post-processing of the interviews sending out the transcripts to the interviewees. We treated the interviewees respectfully and acknowledged their time by pointing out when we reached the planned time frame. We only extended the interview if they offered it. We allowed the interviewees to review the materials and provide feedback before publishing.

3.1.4 Choice of Analysis Method

We applied thematic analysis to build our theory of balancing technological heterogeneity in microservice-based projects. Section 3.2 details the analysis process.

Next to thematic analysis, we considered grounded theory as a competing methodology for data analysis and synthesis. Grounded theory approaches, as described by Strauss and Corbin (1998), act as a framework for generating theories from qualitative data. The approach is predominantly inductive with the goal of creating a theory purely from the data; prior in-depth familiarization with the topic is discouraged to avoid the researchers’ prior knowledge influencing the results. Grounded theory follows a structured coding process. First, the researchers break down data into the initial codes in the *open coding* phase. After, they identify relationships among codes in the *axial coding* phase. Eventually, structured codes are refined into a central category that becomes the foundation of the theory in the *selective coding* phase.

In contrast, thematic analysis, as described by Clarke et al. (2015), is a flexible data analysis method that focuses on identifying patterns of meaning within data rather than on generating a hierarchical theory. Thematic analysis employs a more adaptable coding process. First, researchers generate initial codes, then search for themes, and then finally review and refine the themes; the full process is detailed below. This method supports both inductive and deductive approaches, either deriving themes directly from the data in a bottom-up fashion, similar to grounded theory, or in a top-down fashion, guided by research questions and prior knowledge. The output of thematic analysis is not necessarily a hierarchical, structured theory as in grounded theory; rather, it is a collection of themes that can nonetheless support theory building, offering the researcher considerable flexibility in interpretation and presentation.

The tradeoff described makes thematic analysis a more flexible option. We chose thematic analysis because it aligns better with our research question. Building on our previous research in the field of microservices allowed us to effectively combine deductive and inductive approaches rather than relying solely on grounded theory, which would require all theoretical constructs to emerge inductively from the data.

3.2 Thematic Analysis

We applied thematic analysis, as described by Clarke et al. (2015), to build our theory on balancing technological heterogeneity in microservice-based projects. Thematic analysis is an accessible and systematic research method and procedure to discover, analyze, and interpret patterns of meaning within qualitative data. The researcher takes an active role in generating codes from the qualitative data, guided by the research question. Codes are labeled annotations of text segments that capture interesting features of the data relevant to the research question. Codes are aggregated into themes - patterns of meaning. Underlying is a central organizing concept for the analytic observations.

As primary material for thematic analysis, we used 15 transcriptions of interviews with practitioners (Section 3.1). In an iterative process, we analyzed the primary materials.

We followed an inductive approach, creating themes from data without a predetermined theory, following the six-step process postulated by Braun and Clarke (2006):

1. *Familiarize with the data* We read the primary material actively and noted the first coding ideas. We used the transcription process for the interviews as an excellent way to familiarize ourselves with the interview data.
2. *Generate initial codes* We worked through the primary materials and annotated data segments with preliminary names. We coded as detailed as possible as time permitted and included the context in the coded text segments. Generally, a text segment can be uncoded, coded once, or as frequently as relevant.
3. *Search for themes* We took the long list of codes and considered how differently the codes may be combined. We created the potential themes by aggregating codes that seemed cohesive to us. We thought of relationships between codes and themes and arranged them hierarchically.
4. *Review the themes* We reviewed the themes and codes to reflect on how the individual themes represent the data set. We paid attention to clear distinction criteria of themes and discussed ambiguous ones.

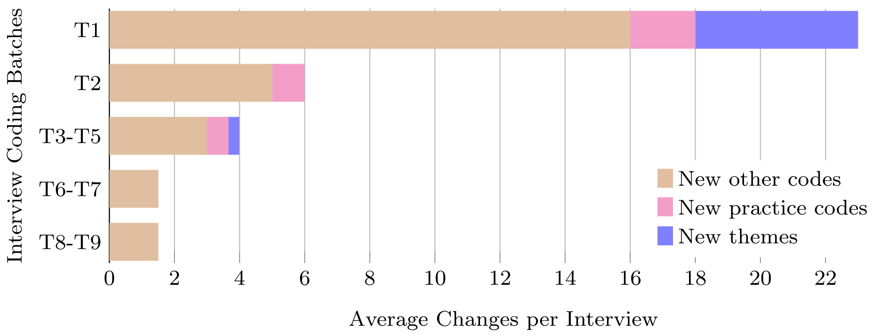

Fig. 2 New insights per interview

5. *Define and name themes* Until now, themes had a working title. We went over each theme individually and identified what is of interest about them concerning the research question and why. We made sure that the themes were not too complex and not too broad by using sub-themes. Additionally, we explicitly put down the definition of each theme and criteria when and when not it applies.

The execution of this process was not as sequential as presented. Moving back and forth between the phases was necessary as we gained new insights during coding and building themes (Braun and Clarke 2006).

To cope with the number of emerging themes, we used the MaxQDA[4] software to support our coding and theme-building process. We attached a memo to each code summarizing its theme in prose and specifying when the code is not applied. The codes, together with these memos, are called the codebook. For reporting, we decided to present codes that were only discussed in a single interview to showcase the diversity of the phenomenon rather than making statistical statements on the relevance of single codes. The exceptions are codes for the best practices where we required evidence from multiple different interviews. The themes' relevance is backed by aggregating multiple codes in a meaningful way.

During the coding process, we periodically saved snapshots to assess the influence of interviews on the code system (after one, two, five, seven, and nine interviews). Figure 2 illustrates the number of themes and codes added per interview. While additional lower-level codes continued to refine the diversity of the phenomenon, like the advantage of technological uniformity simplifying achieving service-level agreements, no new themes or best practices emerged after the first five interviews. We ceased data collection after T9, as further interviews were unlikely to yield novel themes or best practices, which were our core interest to answer the research questions. This decision was based on theoretical saturation, a stopping criterion used in theory building to balance research efforts against the likelihood of gaining new insights.

We conducted two inter-coder reliability sessions to improve the quality and validity of our emerging codebook (O'Connor and Joffe 2020). We chose one secondary rater per iteration that was experienced in the software architecture field to perform investigator triangulation (Guion et al. 2011). The chosen secondary raters applied the existing themes and codes to parts of the uncoded primary materials supported by the codebook entries. The themes, codes, and codebook quality were evaluated qualitatively by comparing the result with the original coding. The secondary raters took notes of missing themes, themes

[4] https://www.maxqda.com/

that need refinement (renaming, redefinition), re-categorization within the themes, and other comments. We discussed the notes jointly afterward to define improvements to the themes and the codebook. Table 2 summarizes the inter-coder reliability sessions. We measured the agreement of the intercoder with the original coder by calculating Cohen's Kappa when coding areas overlap using the features of MaxQDA. According to the interpretation of Landis and Koch (1977), both sessions achieved a moderate agreement with kappa scores of 0.52 and 0.54. After individually checking each coding difference, we trace the moderate agreement back to interview paragraphs with no obvious but rather subtle meanings where the coder's level of knowledge led to different coding decisions. Like in the ”negotiated agreement“ approach of Campbell et al. (2013), we discussed disagreements "in an effort to reconcile them and arrive at a final version in which as many discrepancies as possible have been resolved". This qualitative discussion of all codings showed a reduction in change proposals across sessions, indicating the maturity of the code system. We avoided a learning effect between intercoder sessions by choosing different secondary coders per session.

The results and interpretations were sent to our interviewees, asking them for honest feedback. By conducting a member-checking procedure, we aimed to reduce the risk of potential misinterpretations, omissions, or inaccuracies in our findings (Motulsky 2021). Of the nine interviewees T1-T9, we received three responses within the given timeframe of three weeks. The overall feedback was positive and confirmed our results, with some suggestions on how to improve their presentation. We incorporated their feedback accordingly.

The first feedback included thoughts on how to proceed in their company to work towards a better degree of heterogeneity. In their project, a strong standardization leads to inexperienced teams applying the same mechanisms to every problem. As a result, they see the need to reduce central governance to introduce new technological variations where they are beneficial. Therefore, they acknowledged our set of best practices as a helpful checklist and will evaluate which of them to apply. One open question was how their high-pressure situation to deliver new features would affect the introduction of such new technologies. Further, we received some feedback on the presentation of our theory, such as the use of abbreviations. The second feedback confirmed our findings and gave us an update on their project progress. They introduced a Python microservice recently, with Python being a new technology in their stack. In this context, reading the best practices called forth a feeling of familiarity. The third feedback focused on the style of presentation. They recommended that we improve the use of abbreviations and introduce some overview tables in the results section.

Table 2 Inter-coder reliability

Session		#1	#2
Kappa score		0.52	0.54
Suggestions	Add theme	1	1
	Refine theme	5	1
	Recategorize theme	1	0
	Rename theme	2	0
	Sum	**9**	**2**

4 Results

The result of our study is a theory, abstract knowledge aimed at explaining a phenomenon or predicting outcomes, primarily a set of (typically interrelated) hypotheses rooted in data. We present this theory in the format of a structured research model and an actionable and practitioner-relevant set of 13 best practices.

Figure 3 gives an overview of the built research model by laying out constructs (variables) and hypothesized relationships of balancing technological heterogeneity (TH) in microservice-based industry projects. Following the definition of Palvia et al. (2006), the presented model is a *prescriptive multi-tier influence research model* as we identify different kinds of variables (represented as boxes) and model influencing relationships between them (represented as arrows). We define the independent, intermediate, and dependent variables as follows:

- **Current degree of TH** (independent variable): An n-dimensional record of values (n is the number of considered type of technology (TT)), each as a position on a scale of heterogeneity expressing how heterogeneous the technology choices are in this category; from the same technology used in every microservice on the one end to a different technology in every microservice on the other end.
- **Trade-off effects** (independent variable): Effects of higher or lower degrees of TH, such as the benefits of heterogeneity (BH) and of uniformity (BU) for the project and how they affect the project costs (CP).
- **Environmental factors** (independent variable): Factors (F) of the organizational context that TH decisions are made in, such as developer experience, regulatory boundaries, or project maturity.
- **Optimal degree of TH** (intermediate variable): Similar to the current degree, a degree of TH with n dimensions; it leads to optimal results depending on the prioritization of the trade-offs and the environmental factors.

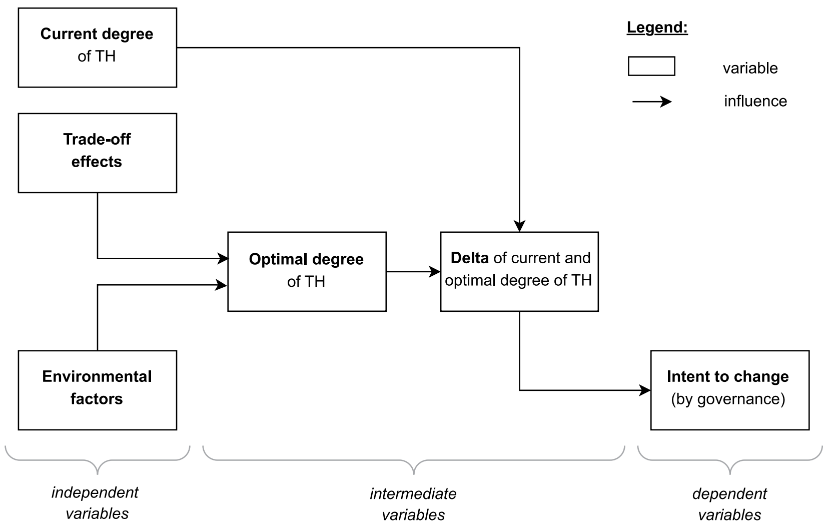

Fig. 3 Research model presenting our theory

- **Delta of the current degree of TH and optimal TH** (intermediate variable): The n -dimensional difference from the current degree of TH to the optimal degree of TH, one value per technological dimension.
- **Intent to change** (dependent variable): The intent is to bring the current degree of TH closer to the optimal degree of TH, minimizing the delta between both states. The intent to change is usually manifested by an active governance process.

The research model describes the decision process to change the current degree of TH and captures the phenomenon at a certain point in time. The result of the intent to change and the actions that are consequently performed lead to a new degree of TH in the future. Although not modeled in Fig. 3, this implies that the current degree of TH is the result of a previous iteration and, thus, is influenced by past environmental factors, past trade-off effects, and the actions taken to implement the intent to change. Introducing and removing technology in projects is a tedious activity, so the iterative nature of the process is well suited to make the endeavor more approachable. Adding or removing technologies one by one allows a gradual migration toward the optimal degree of TH. In addition, environmental factors can change over the course of a project, and so can the optimal degree of TH. Thus, we recommend revisiting the assessment of the degree of TH regularly and viewing the balancing of TH as an ongoing process.

Minimal Example: Consider a project with three microservices, each using a PostgreSQL database. Two microservices are written in JavaScript and one in PHP. The company employs experienced software engineers whose project goal is automated text extraction from PDFs.

Environmental factors are the high experience level of the developers and the project domain of extracting text from PDF using machine learning.
The relevant *trade-off effects* of a higher degree of TH leads to a better technology fit with lower operation costs in the long run on the upside but introduces technological complexity as a drawback. A lower degree of TH leads to opposing effects.
The *current degree of TH* is defined independently for each technological dimension; One technology in three microservices for databases and two technologies in three microservices for programming languages.
The *optimal degree of TH* would be reached by using Python for more efficient machine learning workflows; One technology in three microservices for databases and three technologies in three microservices for programming languages.
The *delta* of adding TH with a third programming language is motivated by choosing a better technology to lower operational costs, while the experienced developers can manage the additional complexity.
From this delta follows an *intent to change* that should be actively managed by governance; Keep database technology the same and add Python as a third programming language.

The following subsections detail the variables and their manifestations. Section 4.1 details the variables *current degree of TH*, *optimal degree of TH*, and *delta of current and optimal degree of TH* by defining the scale of heterogeneity. Section 4.2 describes the *trade-off effects*. Section 4.3 presents four categories of *environmental factors*. Section 4.4 depicts the *intent to change*. Last, Section 4.5 presents best practices (BPs) on how to govern the degree of TH on an organizational level.

4.1 Degree of Technological Heterogeneity (Current, Optimal, Delta)

4.1.1 The Scale of Heterogeneity

The interviews that specifically targeted technological heterogeneity (T1-T9) each reported on their current degree of TH and the experienced effects. We infer from the interviews that there is a scale of heterogeneity in every microservice-based project. The following representative quote from an interview of the set on general integration challenges (G1-G6) describes this scale:

> *"[Technology heterogeneity] behaves like many things. The value curve is [...] a parabola, logically speaking. It has a sweet spot. Too little is not good; too much isn't good either."* - Interview G4, translated from German

On the one end, every microservice uses a different technology, so there is no overlap between microservices. We call this state ”complete heterogeneity“. On the other end, all microservices use the same technology, so there is no diversity at all. We call this state ”complete uniformity“. Both ends delimit a scale where each project can position itself in the current state (see Fig. 4). Further, we can define an optimal state considering the project's environment and the prioritization of trade-off effects. In some projects, a range of practically acceptable goal states might be close to the optimum. For the sake of presentation, we assume that companies strive towards the optimal degree of TH; the insights also find application when striving towards a ”good-enough“ goal state. If the current state is not optimal, then there is a delta that serves as an input to drive change.

As an example, T4 explained that they use Javascript in most microservices except for video rendering services, where they utilize C++ as a programming language. In terms of programming languages, this positions the project closer to the uniform side, not at the extreme, though, where one and only one programming language is used.

Summary: Technological uniformity and technological heterogeneity are the two ends of a scale of an identifying value of heterogeneity for a given project. We call this value *the degree of TH*. Even without active management, projects always land somewhere on the scale for their current state. Additionally, projects can define a (range or a single) goal state close to the optimal state to which they aim to converge.

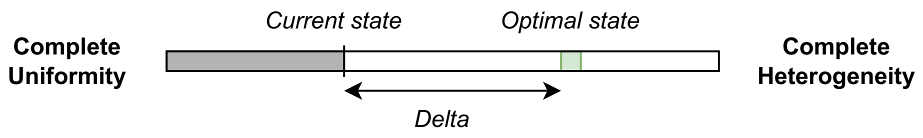

Fig. 4 Technological heterogeneity vs. uniformity

4.1.2 One Dimension per Technology Type

Programming languages are only one technological dimension where such a trade-off occurs. Table 3 gives a complete list of technological dimensions that the interviewees discussed, each covering related types of technologies (TT). We distilled the following categories of technological dimensions:

– Application technology: programming languages, frameworks, libraries, databases (TT1, TT7, TT6, TT3)
– Deployment technology: CI/CD pipelines, deployment platform, service configuration (TT2, TT4, TT12)
– Cross-cutting aspects technology: microservice APIs, logging and monitoring, documentation, testing, user interface, authentication (TT5, TT8, TT9, TT11, TT10, TT13)

> *"So in the language area, I think we're quite strict. [...] if someone goes into introducing a cache, I think it would be more flexible. If we think about long-term storage and so on, then a little bit more strict as well due to our regulations [...] It's really standardized, and our services communicate with each other in the same manner."* - Interview T2

We discovered that the technological dimension influences the degree of TH projects strive for. The implications of diversity in some types of technology vary significantly. For example, deploying to different deployment platforms will impact the daily development and operation efforts more than using different technologies to document the microservices.

Table 3 Technology types

	Technology Type	Interviews
TT1	Programming language	T1, T2, T3, T4, T5, T6, T7, T8, T9
TT2	CI/CD pipeline	T1, T2, T3, T4, T5, T6, T7, T8, T9
TT3	Database	T1, T2, T3, T4, T5, T6, T7, T8, T9
TT4	Deployment platform	T1, T3, T4, T5, T6, T7, T8, T9
TT5	Microservice APIs	T1, T2, T3, T4, T5, T7, T8, T9
TT6	Programming library	T2, T4, T5, T6, T7, T8, T9
TT7	Programming framework	T4, T5, T6, T7
TT8	Logging and monitoring	T1, T2, T7, T8
TT9	Documentation	T2, T4, T7
TT10	User interface	T3, T6, T7
TT11	Testing	T2, T8
TT12	Service configuration	T6
TT13	Authentication	T9
TT14	Quality criteria of microservice*	T2, T3, T8, T9
TT15	Internal structure of microservice*	T1, T2, T4, T7
TT16	Coding style*	T7
TT17	Cloud resource ownership*	T1

We conclude that the TH of a project can be classified along n axes, where n is the number of technological dimensions considered. For each technological dimension, the trade-off between heterogeneity and uniformity can be made. For example, a project may use one uniform CI/CD pipeline technology and only two selected API technologies for microservices while using a diverse set of programming languages and databases.

Notably, some interviewees also reported heterogeneity considerations on non-technological subjects, indicated by a * in Table 3. The most frequent representatives are non-functional quality criteria of microservices (TT14), like adhering to security standards ensured by vulnerability scanning and the internal structure of microservices (TT15). Code styling (TT16) and ownership of cloud resources (TT17) fall into the same non-technology bucket.

Summary: The degree of TH may vary depending on the technological dimension. A project can be uniform in one technological dimension (e.g., the programming language) but heterogeneous in another (e.g., the storage technologies).

4.2 Trade-off Effects

4.2.1 Benefits of Both Sides

The benefits of technological heterogeneity and uniformity pull to both ends of the scale. Each benefit of one side poses a drawback to the other. Table 4 lists four benefits of technological heterogeneity (BH) and seven benefits of uniformity (BU) the interviewees reported. In the table, the desirable trade-off effect is boldfaced for each benefit.

> *" Advantages are [...] that you are free to choose what is best for the microservice's question. For instance, we have one microservice which is based or written in Python as it's using machine learning and all the machine learning libraries. [...] It's kind of natural for having this language there.*
>
> *[...]*
>
> *[The service written in Python], I think it's the one who's usually causing some monitoring issues. [...] But also, there's nobody in our company [...] who actually knows Python or the respective REST API framework. [...] I think it's more an advantage than a disadvantage to have a uniform programming language. "* - Interview T1

Using these best-fitting technologies (BH1) and learning which technologies work best in specific scenarios (BH2) are the primary motivations for favoring technological heterogeneity. The main drivers for technological uniformity are simplified maintenance (BU1), code reuse (BU2), reducing the impact of knowledge barriers and entailed learning efforts in terms of onboarding in new teams and code bases (BU3), moving engineers between teams from a capacity planning perspective (BU4), becoming experts in the used technology (BU5), and being able to contribute to other services occasionally (BU6).

We found that whether heterogeneity saves or causes costs is more complex than a linear relation. Thus, we devote the next subsubsection to explaining the effect on costs in a project separately.

Table 4 Trade-off between technological heterogeneity and uniformity

	Characteristic	Complete Uniformity	Complete Heterogeneity	Interviews
BH1	Ability to use best fitting tech	low	**high**	T1, T2, T5, T6, T7, T8, T9
BH2	Knowledge which tech works best	low	**high**	T3, T9
BH3	Coupling by technology	high	**low**	T2
BH4	Technology lock-in	high	**low**	T4
BU1	Complexity of maintenance	**low**	high	T1, T3, T5, T7, T8
BU2	Code reuse	**high**	low	T1, T2, T5, T6, T9
BU3	Onboarding and learning efforts	**low**	high	T1, T4, T6, T7, T8
BU4	Ability to move engineers betw. teams	**high**	low	T2, T3, T5
BU5	Potential to build deep knowledge	**high**	low	T4, T5, T6, T9
BU6	Ability to contribute to other services	**high**	low	T4, T7
BU7	Complexity to achieve SLAs	**low**	high	T9

The desired manifestation of each characteristic is **boldfaced**

Weighing the benefits of technological heterogeneity against the benefits of technological uniformity depends on the project's context. For example, choosing the best tool for the job may outweigh all the benefits of uniformity in a scenario where an inappropriate technology makes an implementation infeasible or inefficient. Other projects might favor maintainability by a uniform technology stack over all the benefits of technological heterogeneity.

Summary: Finding the right trade-off means finding the right balance between (a) a good technology fit and long-term flexibility and (b) the more complex maintenance, the lack of code reuse, and increased efforts for onboarding and people management. The prioritization of benefits impacts the optimal degree of TH.

4.2.2 Costs of Introducing Technology

Besides the presented benefits of technological heterogeneity and uniformity, we found a complex relationship between introducing new technology and costs for the project (CP). To understand which side of the trade-off costs favor, we asked the interviewees about the relationship between the degree of technological heterogeneity and the project costs. Our

Table 5 Costs effects of heterogeneity

	Characteristic	Complete Uniformity	Complete Heterogeneity	Interviews
CP1	Short-term costs (tech introduction)	**none**	some	T1, T2, T3, T4, T5, T6, T9
CP2	Long-term costs (from tech fit)	high	**low**	T1, T2, T3, T5, T6, T8
CP3	Maintenance cost	**low**	high	T1, T5, T7, T8, T9
CP4	Operation cost	**low**	high	T1, T2, T4, T5, T6, T7, T9
CP5	Governance cost	**low**	high	T2
CP6	Learning/training cost	**low**	high	T2, T5, T6, T7, T8
CP7	Complexity of cost estimation	**low**	high	T1, T2
CP8	Hiring cost	high	**low**	T5
CP9	Costs of technology lock-in	high	**low**	T4, T9

The desired manifestation of each characteristic is **boldfaced**

findings in Table 5 suggest that the degree of technological heterogeneity does not linearly correlate with the costs it causes.

> *"I think if you stay non-heterogeneous, uniform, then at least the cost is easy to calculate. But on the other hand, you can also get to a situation where you never get the benefits of something new. So at least taking a look at new technologies, there's always reasons why somebody invents new mechanisms. Usually, the reason is that it makes development faster or easier. And I think changing the technology is always an investment. But every investment also has a return of investment."* - Interview T1

On the one hand, a well-chosen technology can save costs in the long run by using more efficient and sustainable technology (CP2). Heterogeneity may decrease hiring costs since a larger pool of applicants is available (CP8). Relying on multiple technologies can prevent coupling to technology and reduce the costs of technology lock-in (CP9).

On the other hand, each introduction of further technology causes increased short-term costs (CP1), but also maintenance, operation, governance, and learning and onboarding costs (CP3-CP6). Furthermore, uniformity simplifies cost estimation (CP7).

Summary: Introducing new technology imposes short-term costs on the project with the potential to save costs in the long run by an increase in efficiency. The prioritization of different types of costs impacts the optimal degree of TH.

4.3 Environmental Factors

The direction to which a project naturally leans and where the optimum is located depends on its environment. We distilled four categories of environmental factors that influence how the benefits of technological heterogeneity on the one side and uniformity on the other are weighed against each other.

The **software engineers' skills (F1)** provide a foundation for what technologies can be adopted efficiently. If there is existing expertise in a technology that is about to be introduced, the efforts of introducing and operating the technology (BU3, CP4, CP6) decrease significantly. If engineers are proficient with multiple technologies of the same type, the advantage of uniformity of moving them between teams becomes less significant (BU4).

> *"It's more the question, what expertise do we have? And can we afford to add another technology where only very few people are aware of it? So there is a pressure to use the technology where we have a critical mass of staff who knows how to deal with that technology [...]"* - Interview T8

The **culture of the company and the project (F2)** decide how additional introduced technology will be accepted and who the driver for technological innovation is. The company's age, size, and maturity lead to different fundamental values that we summarize in this influence factor. A culture generally accepting new technology lowers onboarding and learning efforts (BU3, CP6). Bureaucratic processes and hierarchical structures can affect the governance costs of technology (CP5) and the short-term costs of introducing new technology in general (CP1). *"At some point, the size becomes a bottleneck in the sense of too many voices. [...] Because you have too many voices and too many different opinions, and then either each team does whatever they want, and you don't have any consensus and then you lose this flexibility that I mentioned before about switching resources. And thus we need a governance body or a very top-down approach to even enforce techs onto the teams."* - Interview T5

The **project phase (F3)** determines how much resources are spent on introducing and maintaining additional technology in the context of balancing the agility and stability of the project. The priorities for spending the budget change over the course of a project. In the beginning phases of a project, more budget is planned for programming features, while it shifts to maintenance efforts over time. Thus, introducing new technology can be perceived as worthwhile in the early stages of a project as the impact of certain types of costs on the overall budget changes. *"If you want to be very fast, if you want to release new features like every couple of days, for example, and if, basically, speed is the thing that defines that you should give independence or the freedom of choice to the teams. If costs are more of a driving factor, you will focus on cost-efficient designs. There can also be a shift in focus from speed of innovation towards reduction of cost."* - Interview T9

The **project's domain (F4)** can impose restrictions on how heterogeneous technologies can be and which ones are available in the first place. For example, regulated industries like finance or healthcare require specifically certificated technologies and limit the pool of available technology. This boundary condition limits the possible technology fit (BH1, CP2) that heterogeneity can achieve. *"[...] if you're looking at the financial sectors, insurance companies, they are a bit narrow with the technologies used. It's gaining more and more speed to introduce new technologies. It just takes a bit more time. Nevertheless, you see a couple of different databases. Compared to the younger industries like e-commerce and online platforms, for example, they introduce technologies more frequently and you see a wider range of databases and a wider range of programming languages. So there is a difference depending on how regulated the industries are [...]."* - Interview T3

Summary: The environment of a project impacts which technologies are introduced naturally, leading to the current state of TH. Further, the environmental factors influence how the benefits of both sides of the trade-offs and the individual costs of introducing a technology are perceived and weighed against each other, impacting the optimal degree of TH.

4.4 Intent to Change

The delta of the current to the optimal degree of TH leads to an intent to change. In practice, this delta must be large enough to trigger action. The long-term general and financial benefits of changing the current degree of TH must outweigh the initial costs and general drawbacks of introducing/maintaining or removing technology.

We logically infer that the reason technology is introduced contributes majorly to the profitability of the decision. Similarly to Brooks and Bullet (1987) who introduce the concepts of essential and accidental difficulties in software engineering, we distinguish between essential and accidental heterogeneity.

If technology is or was introduced only for technology's sake, e.g., because the technology is fun or the engineers' favorite, then the benefits of the increased heterogeneity will likely not outweigh the costs because the benefits cannot be actively reaped. We call this kind of diversity *accidental heterogeneity*. The technology choices then exceed what is necessary to encompass the software domain's unique requirements, constraints, and intricacies - making the technology landscape more complex than it has to be.

If, however, technology is or was introduced with a clear goal in mind, e.g., to increase development or operation efficiency, then the benefits of the technology have a high probability of outweighing its costs. We call this kind of introduced diversity *essential heterogeneity*. The technology choices then serve to address the software domain's subtleties effectively and efficiently without adding more complexity to the technology landscape than necessary.

If no goal state can be worked against, accidental heterogeneity can be introduced to the system, significantly affecting the project's costs.

"I think [...] the governing aspect of it is almost more important than the technology itself. [...] so far where I have experienced [...] the most friction or complications is almost more in the organizational side of it than in the technological side of it." - Interview T4

Technologies should be governed in a project to avoid the entry of accidental heterogeneity. Section 4.5 presents best practices to govern the technological heterogeneity of a project.

Summary: Accidental technological heterogeneity emerges by introducing technology for technology's sake, while essential technological heterogeneity emerges by introducing technology with a good reason. Technologies should be managed to avoid accidental TH and to encourage essential TH.

4.5 Governance Best Practices

We distilled 13 industry best practices (BPs) from the interviews that can guide the intent to change. We present each best practice in the classical pattern format with context, problem, and solution. We choose this form to easily match a specific problem situation to the part of our theory and explain how to solve the problem (Riehle and Züllighoven 1996). This presentation is optimized for practitioner relevance and ease of access. We intentionally refer to them as *best practices* rather than *patterns* to emphasize that they result from a research methodology, grounded in the insights provided by the interviewees. In future work, these best practices can be expanded to patterns with repeated practical observation or development through workshops, as is common in the pattern community.

Figure 5 provides an overview of the contents of this section and outlines the presented practices and their interconnection. Standardization is the core governance tool to balance the trade-off between technological heterogeneity and uniformity. Standardization captures the consensus of all or most microservice teams on a particular aspect.

BP1: Meaningful standardization
Context:
There are multiple cross-functional teams, each responsible for one or more microservices.
Problem:
(1) The diversity of technology (tech stacks, programming languages, frameworks, data stores) leads to additional costs, and limits the effectiveness of moving people from one team to another due to high knowledge barriers.
(2) The stringent enforcement of one specific technology stack limits the flexibility and introduces complexity by using not suiting tools.
Solution:
Introduce meaningful standardization to welcome essential technological heterogeneity and prevent accidental one. Establish processes and mechanisms to (i) create standardization, (ii) disperse the knowledge of their existence and how to implement them across the microservices teams, and (iii) consistently apply them.
Sources of evidence:
T1, T2, T3, T4, T5, T6, T7, T8, T9, G1, G2, G4, G5, G6
Representative quotes:
"It makes sense to define a set of minimal standards so that [the microservice teams] don't have to start at zero when coordinating, so that coordination can be reduced to the minimal amount that is necessary. It is a trade-off at the end of the day. [...] The value curve is kind of a bell-shaped curve or a parabola, logically speaking. It has a sweet spot. Too little is not good; too much isn't good either." - G4, translated from German

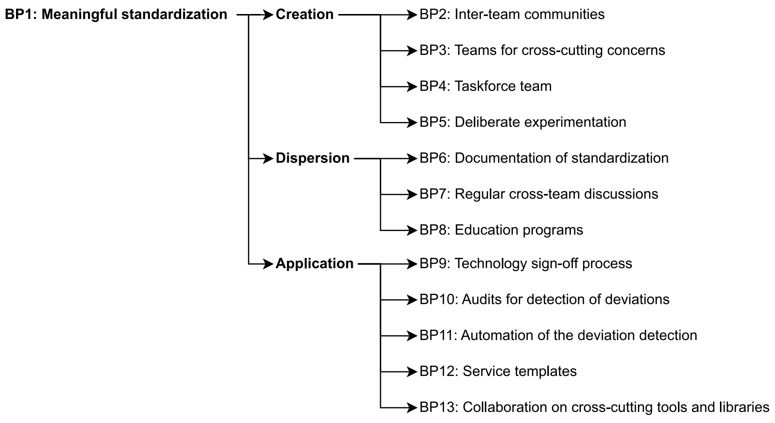
BP1: Meaningful standardization
Creation
BP2: Inter-team communities
BP3: Teams for cross-cutting concerns
BP4: Taskforce team
BP5: Deliberate experimentation
Dispersion
BP6: Documentation of standardization
BP7: Regular cross-team discussions
BP8: Education programs
Application
BP9: Technology sign-off process
BP10: Audits for detection of deviations
BP11: Automation of the deviation detection
BP12: Service templates
BP13: Collaboration on cross-cutting tools and libraries

Fig. 5 Best practices overview

Establishing **BP1: meaningful standardization** ensures that no accidental technology is introduced. Thus, it reduces the impact of the drawbacks emerging by technology becoming too heterogeneous by chance. Standardization may describe multiple agreed-on technology stacks that serve as alternatives for different use cases. In this way, standardization allows for essential and meaningful heterogeneity that enables drawing benefits from technological heterogeneity while limiting its drawbacks.

For many subjects, focusing the standardization on an interface rather than the underlying technology makes sense. This allows a free choice of technology as long as the interface is implemented. Exemplary candidates to standardize the interfaces over technology are:

– The communication protocols used between microservices (e.g., HTTP) over the programming languages and frameworks used within the microservices (e.g., Java);
– The interfaces for collecting logging, monitoring, and tracing data (e.g., logging to stdout stream) over libraries used for the implementation (e.g., Log4j);
– The deployment artifact each microservice produces to allow uniform deployment to a common deployment platform (e.g., container images to deploy to a Kubernetes cluster) over a script to build the deployment artifact optimized for a specific deployment platform (e.g., bash scripts to deploy to a Glassfish application server).

Deciding whether to standardize the interface rather than the implementing technology depends on the project context. The best practices presented in the upcoming subsections apply to both scenarios.

To create and maintain meaningful standardization, we structured the best practices into three major activities. First, meaningful standardization needs to be actively created. Second, the awareness of the existence of created standardization needs to be dispersed across the organization. Executing entities must be aware of standards and build the required know-how to implement them. Third, the consistent application of the standardization needs to be ensured. Dedicated organizational and technical mechanisms can enforce or incentivize the consistent application of standardization.

The following subsections detail how to create standardization (Section 4.5.1), how to disperse knowledge and awareness for such across an organization (Section 4.5.2), and how to ensure the consistent application of standardization (Section 4.5.3). For readability purposes, we highlight one best practice per chapter. The complete list of best practices is available in Appendix A.1.

4.5.1 Creation of Standardization

This set of best practices is applicable when multiple cross-functional teams are responsible for one or more microservices, and **BP1: meaningful standardization** shall be introduced.

However, there is a lack of knowledge to select suitable technology for such standardization, or the standardization may lack the backing of the microservice teams.

Standardization can emerge from various groups in an enterprise environment. We found the following organizational units to suit the creation of standardization (see Fig. 6).
If the standardized technology is questioned or even partially ignored by the microservice teams, **BP2: inter-team communities** can take responsibility for the creation of standardization. Such communities are composed of delegates of the microservice teams, often brought together by a similar role in their team or their interest in certain topics. For example, each team sends their architecture-savvy person to the architecture community or their security-savvy person to the security community. The communities can be organized democratically or be chaired by a global role, like the global architect. Even though the chairperson might have the final word, every team can participate in the discussions. In addition to general coordination, those communities can be responsible for defining standardization within their topic of expertise. By involving the microservice teams in this manner, the standardization has a higher chance of being backed by the individual teams.

BP2: Inter-team communities

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1).

Problem:

Global decisions for standardization lack the backing of the teams, are questioned, or are even partially ignored.

Solution:

Establish inter-team communities for decision-making, like standardization. Communities revolve around certain subjects, usually around a cross-cutting concern or a technology type. Subject-savvy and interested members of the different microservice teams can participate in the communities and represent their team in the community. The community (a) keeps an overview of their topic, (b) makes critical decisions, (c) assigns the responsibility of shared capabilities or bears it itself, and (d) disperses information across teams by their representatives.

Sources of evidence:

T1, T2, T4, T5, T6, T7, T8, T9, G2, G3, G4, G5, G6

Representative quotes:

"We like to speak of micro and macro architecture or micro and macro decisions. Micro is everything I can decide on my own. Macro would be something like the API design guide [...], or a security concept, or which tracing infrastructure and format to support. [...] these are macro decisions that someone has to make. Usually a group. What we frequently do is that every service team sends an architecture-savvy representative to an architecture board. And there exactly these decisions are made [...]." - G2, translated from German

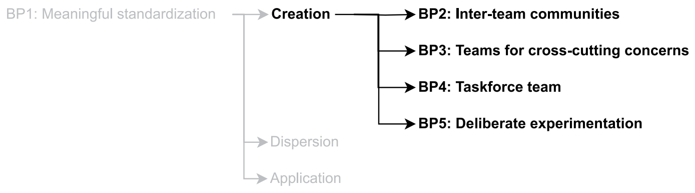
BP1: Meaningful standardization
Creation
BP2: Inter-team communities
BP3: Teams for cross-cutting concerns
BP4: Taskforce team
BP5: Deliberate experimentation
Dispersion
Application

Fig. 6 Best practices for standardization creation

If the additional load on the microservice teams caused by the need to use the standardized technology is an issue, introduce **BP3: teams for cross-cutting concerns** as organizational units. Such a team takes responsibility for a cross-cutting concern and, thus, is a natural fit to define standardization and support the individual microservice teams adhering to the standardization. For example, a monitoring team takes full responsibility for operating the monitoring infrastructure and defines the logging and monitoring formats each microservice has to implement. Such entities are particularly suited for standardization creation when technology can be maintained and operated as a shared component.

If the microservice teams cannot live up to the knowledge requirements to use the standardized technology, establish **BP4: a taskforce team** as a front-runner team. Its job is to evaluate and introduce technologies across the whole architecture, like the monitoring infrastructure described beforehand. Afterward, they support the individual microservice teams in coping with the challenges and act like an internal consultancy unit. Compared to teams for cross-cutting concerns, a taskforce team does not take long-term responsibility for shared components but moves on to tackle new challenges. A taskforce team evaluates suiting technology in this process and may introduce systematic standardization. The interviews suggest that such a taskforce team is a generic best practice to tackle various challenges of introducing a microservice-based architecture beyond purely introducing technology.

If the currently standardized technology does not satisfy all use cases efficiently, consider **BP5: deliberate experimentation**. Allowing for experimentation phases is a planned approach to discovering new technologies and evaluating their fit and whether they are worth the cost to introduce them across the architecture. Experimentation can occur globally to introduce a new technology to the standardization or in a dedicated taskforce team. Still, it might also occur locally within a microservice to solve a specific problem and then be adopted for standardization afterward.

4.5.2 Dispersion of Standardization

This set of best practices is applicable when multiple cross-functional teams are responsible for one or multiple microservices and first efforts were made to introduce **BP1: meaningful standardization** by the **creation of some project-wide standards** (Section 4.5.1). However, the information on standardization is not flowing well to the individual microservice teams. The awareness and knowledge of standards is crucial. Without, there can’t be a consistent implementation across the project.

On the one hand, this information can be dispersed by the units responsible for creating the standardization (see Section 4.5.1). On the other hand, there might also be more sophisticated dispersion mechanisms like the following best practices (see Fig. 7).

BP6: Documentation of standardization and its decision process can serve as a look-up for the microservice teams on the current standardization. Such documentation should include an allow-list of technology that can be used without discussion. Maintaining the standardization documentation in one place allows easy access and improves its discoverability.

BP6: Documentation of standardization

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1). A standardization has been created or updated.

Problem:

The information of standardization decisions does not flow to the microservice teams which should adhere to the standardization.

Solution:

Document the standardization at a defined place. Documenting not only the outcome but also the considerations surrounding technology decisions serves as a look-up for the microservice teams. Developers can inspect which technology is usable without any discussion. Organizing all standardization documentation in one location instead of dispersing it to multiple locations improves discoverability.

Sources of evidence:

T1, T2, T4, T5, T9, G2, G5

Representative quotes:

"But this was documented then in the end of the session in the Confluence [wiki] page. And there was also a complete tech stack page that every engineer could look up and just see what were green-lighted techs that they could use already without discussion." - T5

If situations arise that point to a lack of knowledge transfer between teams in general, **BP7: regular cross-team discussions** can be utilized to announce and present new aspects of the standardization besides presenting the teams' progress and general insight. Teams may send representatives who return the knowledge to their teams instead of attending with all team members. Introducing such regular meetings can be as simple as reserving a fixed time slot and offering a stage for the different microservice teams.

If there is a lack of knowledge on how to implement some standardization, internal or external **BP8: education programs** can further coach microservice teams to implement the standardization correctly. Such education may come in different formats, like lectures, workshops, blueprints, or tutorials.

4.5.3 Application of Standardizations

This set of best practices is applicable when multiple cross-functional teams are responsible for one or multiple microservices and efforts were made to introduce **BP1: meaningful standardization** by the **creation of some project-wide standards** (Section 4.5.1) and the **dispersion of knowledge about these standards** (Section 4.5.2). However, the standardization is not consistently applied across all microservice teams.

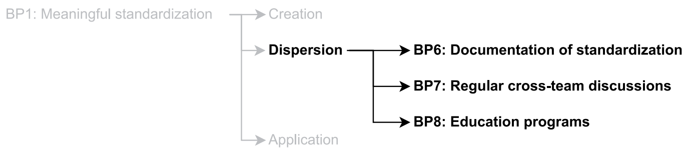
BP1: Meaningful standardization
Creation
Dispersion
BP6: Documentation of standardization
BP7: Regular cross-team discussions
BP8: Education programs
Application

Fig. 7 Best practices for standardization dispersion

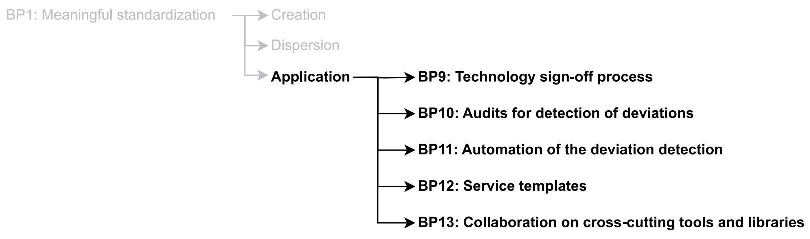
BP1: Meaningful standardization
Creation
Dispersion
Application
BP9: Technology sign-off process
BP10: Audits for detection of deviations
BP11: Automation of the deviation detection
BP12: Service templates
BP13: Collaboration on cross-cutting tools and libraries

Fig. 8 Best practices for standardization application

On the one hand, the dispersed awareness and knowledge of standardized technology will improve standards adoption (see Section 4.5.2). On the other hand, there can be more sophisticated mechanisms to enforce or incentivize adherence to the standardization. Enforcing mechanisms entail adhering to the standardization becomes mandatory for every microservice team. Incentivizing mechanisms encourage adhering to the standardization but do not enforce doing so. We found the following best practices to apply standardization consistently (see Fig. 8).

To enforce the standardization, a mandatory **BP9: technology sign-off process** can be introduced that requires explicit approval of not yet utilized technology. This process empowers a central organizational entity to keep an overview of the development of the technology landscape and actively govern it. Introducing the need to justify additional technology in front of a technology committee and getting their approval as additional effort can significantly reduce the amount of accidental technology in a system.

In addition, **BP10: audits for the detection of deviations** from the standardized technology can be utilized. Regularly reviewing the technologies the individual microservice teams use helps to detect accidentally introduced technology. Such audits can be implemented as parts of existing review processes, like code reviews. Especially in regulated industries like finance or health care, audits are a commonly employed regulatory process and are a natural fit to ensure only standardized technologies are used. Such domains might pose additional restrictions on technology; for example, they might require certain certifications for the technologies used.
Complementary or as an alternative to audits, the **BP11: automation of the detection of deviations** from the standardized technology can be strived for. Software tests can be implemented and added to Continuous Integration pipelines by making the standardization measurable. Depending on the implementation of whether a warning or an error is raised on a deviation, the mechanism can serve for enforcement or incentivization. Including additional context information on the standardization can also serve as a communication tool to disperse required knowledge.

BP11: Automation of the deviation detection

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (BP1). A standardization has been created or updated. The microservice teams are aware of that and have the knowledge to adhere to the standardization

Problem:

Some microservice teams do not adhere to the standardization.

BP11: Automation of the deviation detection

Solution:

Automate the detection of deviations from the standardization, e.g., as software tests in the CI pipeline. Making the standardization measurable and bringing it to the attention of the developers enforces or incentivizes adherence to the standardization, depending on whether the CI pipeline fails or just logs a warning. If furnished with a clue on how to implement the standardization, such automatization can also serve as a communication tool to disperse knowledge.

Sources of evidence:

T1, T2, T4, T9, G1, G2, G6

Representative quotes:

"Similar to if you use Sonar cloud for checking your code quality. [...] So having like a measurable standard. All automated for sure. That would help a lot. That you have this one dashboard where it says, okay, these two services are running behind the standard. [...] So I would not enforce but rather kind of point out that the standard is violated. With a clear indication on how to fix it." - T1

To incentivize the use of standardized technology, **BP12: service templates** can be introduced. A service template serves as a starting point for new microservices with all the required standardization built in. The convenience of a quick start into a new microservice like this incentivizes using the standardized technology rather than spending efforts to create a different microservice seed.

Complementary or as an alternative to service templates, **BP13: collaboration on cross-cutting tools and libraries** can incentivize using the standardized technology. Enabling the reuse of existing tools and libraries poses an incentive to use standardized technology instead of re-implementing an equivalent solution. Since the use cases of microservices might slightly differ, adopting an open-source-like process within the company (called inner source) can foster a seamless evolution of such shared tools.

5 Discussion

In this section, we discuss the implications of our findings. Section 5.1 describes how practitioners can use our results as a practical guide. Section 5.2 discusses the implications of our findings in the field of microservice research.

5.1 A Guide for Practitioners

Section 4.5 presents the best practices we derived from the underlying interviews via thematic analysis, a method to ground a research theory in data meticulously. Still, practitioners might want further guidance that exceeds the presented research model and best practices. Thus, we intend to showcase how the best practices can be employed in combination over the lifetime of a project depending on the influential factors we carved out in Section 4.

This subsection complements the empirical data by incorporating our broader observations from extended engagement with industry practitioners beyond the presented interviews. As such, it offers an informed but interpretative guide for practitioners, synthesizing key insights into actionable recommendations. However, as these recommendations are not derived directly from the collected data, future research should further validate their applicability. Table 6 gives an overview of the underlying hypotheses that can be validated in future work. We added more fine-grained hypotheses that complement the higher-level ones. This is not a complete list of hypotheses, but rather a selection we expect to be the most interesting and have the highest chances of holding true in validation studies.

Table 6 Hypotheses for future validation studies underlying the practitioner guide

#	Hypotheses
H1	The project phase has a significant impact on the optimal degree of TH.
H1.1	Early-stage projects benefit significantly less from standardization (BP1).
H1.2	Mid and late stage projects benefit significantly more from standardization (BP1).
H1.3	Using cross-functional discussions (BP7) in early-stage projects reduces the degree of TH without explicit governance of technologies.
H2	The project has a significant impact on the degree of TH.
H2.1	The naturally emerging degree of TH without active governance is significantly higher in projects with a culture that values technology openness compared to those that value established technology.
H2.2	The optimal degree of TH is significantly higher in projects with a culture that values technology openness compared to those that value established technology.
H2.3	Service templates (BP12) are adopted significantly better in projects with a culture that values technology openness compared to those that value established technology.
H2.4	Deliberate experimentation (BP5) has a significantly greater effect in projects that value established technology compared to those that value technology openness.
H3	The skill level of software engineers has a significant impact on which best practices have the most positive impact.
H3.1	Projects with inexperienced software engineers benefit significantly more from taskforce teams (BP4) compared to projects with very experienced software engineers.
H3.2	Project with inexperienced software engineers benefit significantly longer from deliberate experimentation (BP5) compared to projects with experienced software engineers.
H4	The project domain has a significant impact on the degree of TH.
H4.1	Projects in regulated domains (e.g., healthcare) lead to a significantly lower naturally emerging degree of TH without governance.
H4.2	Projects in regulated domains (e.g., healthcare) lead to a significantly lower optimal degree of TH.
H4.3	Projects in unregulated domains benefit significantly more from best practices that incentivize the application of standardization (BP12, BP13) compared to ones that enforce standardization (BP9, BP10, BP11).
H4.4	Projects in regulated domains benefit significantly more from best practices that enforce the application of standardization (BP9, BP10, BP11) compared to ones that incentivize standardization (BP12, BP13).

5.1.1 Influence of the Project Phase (F3)

In our experience, early-stage projects often prioritize moving fast and breaking things over maintaining the stability of the existing software. Thus, we advise being more open to heterogeneity in the early stages of a project, which allows learning which technology works well (BH2) for the teams and the project domain. It has proven beneficial to introduce the obligation to discuss technological decisions in the form of **BP7: regular cross-team discussions** soon. A frequent exchange of the teams' learnings has the potential to naturally limit the choice of used technology as new microservices will tend to be inspired by what is already used in the project and has proven to work well. In cases where it is technically necessary, e.g., using a common frontend technology for a monolithic frontend, a standardized technology should be chosen by involving all teams in the discussion.

Once a project becomes more mature, favoring stability and maintainability over pure implementation speed, we advise spending efforts on aggregating which technologies are used and consolidating technology by introducing **BP1: meaningful standardization**.

5.1.2 Influence of the Culture (F2)

Culture represents a set of shared opinions and thought patterns that exist throughout a company or a project. Those similarities make coordination more straightforward, as some things are just taken as given. On the other hand, culture is difficult to change once established.

If a project's culture values openness to new technology, the technology landscape will automatically become more heterogeneous. Less effort must be put into technology innovation as it comes naturally, while more efforts must be put into ensuring heterogeneity doesn't get out of hand, e.g., by incentivizing certain technologies with **BP12: service templates**.

If a project's culture values established and known technology, there is a high probability that the microservice teams will stick to one technology stack by themselves. Less effort must be put into technology management as uniformity comes naturally, while more efforts must be put into ensuring technological innovation within the project, e.g., by **BP5: deliberate experimentation**.

5.1.3 Influence of the Software Engineers' Skills (F1)

If engineers are relatively inexperienced with the plethora of technology required to implement microservices, it might make sense to introduce a **BP4: taskforce team** to have a front-runner team that builds up the required knowledge. A fair amount of **BP5: deliberate experimentation** will be necessary to find a well-suited technology stack, but also to overcome the architectural challenges of microservices in general.

If the microservice teams are composed of experienced software engineers, a taskforce team might be of less value. Experimentation still has its place but is less exercised in a more targeted way on specific topics rather than applied broadly.

5.1.4 Influence of the Project Domain (F4)

While projects in non-regulated domains may be intrinsically motivated to standardize technology, projects in regulated domains need to pass audits and only use certified technology. This serves as additional extrinsic motivation to establish standardization.

In domains without regulation, we advise incentivizing the consistent application of standardized technology over enforcing it. Introduce a **BP12: service template** and encourage **BP13: collaboration on cross-cutting tools and libraries** to simplify sticking to the primary technology stack. The teams should be empowered to deviate from the standardization, but be aware that they add additional responsibilities to their workload. These incentives allow the flexibility to choose better tools where it makes sense while making it an explicit consideration by the additional effort.

In regulated domains, technologies should be managed more strictly, e.g., by a **BP9: technology sign-off process** and conducting **BP10: audits for the deviation detection**. Still, we advise involving the microservice teams in technology decisions rather than putting them under strict top-down management to improve the acceptance of technology decisions.

5.1.5 Context-Independent Aspects

Once you start with the processes of introducing meaningful standardization to a project, introduce a **BP3: team for the cross-cutting concern** of operation, including logging and monitoring. This team will take the weight of taking care of the deployment platform and operational aspects from all microservice teams. The operation team may settle on standardized interfaces rather than concrete technology to ensure compatibility with the majority of technology in the microservices and support the microservice teams in working towards a stable deployment pipeline.

For other aspects, like architecture or security, we advise introducing **BP2: inter-team communities** chaired by a lead architect or a lead security engineer. The communities should engage in discussion and settle on a main technology stack in terms of programming language, framework, and storage technology to express the main business logic. In addition, there might be additional technology stacks for special use cases where the primary technology stack is not a good fit, such as machine learning or image processing.

All decisions for and against technologies should be documented in a central **BP6: documentation of standardization**, serving as a single source of truth on which technology is contained in the primary technology stack. An active communication channel should be established, e.g., via email, where changes to the technology standardization are announced and **BP8: education programs** in the form of blueprints and tutorials are dispersed.

Even after a project matures and aggregates more maintenance obligations, it is crucial not to stop innovating technology. Microservices are small units that promise to be easily replaceable, fostering technological innovation and gradual migrations. Keeping the **BP4: taskforce team** around might make sense to focus on **BP5: deliberate experimentation**. However, this only works for larger projects with higher budget, since maintaining a team solely for experimentation is expensive. An alternative approach we experienced at one of our industry partners was allowing a fixed percentage of a developer's time to project-related side-topics, which may include evaluating promising technologies for specific use cases.

5.2 Organizational Challenges as Research Opportunities

We initially started out in a previous study on integrating microservices in general, expecting to come across various technical challenges. However, the results showed that solutions for technical challenges exist and are well known, while companies struggle more with organizational ones. This study devotes itself to one of such challenges - balancing the trade-off of technological heterogeneity.

While topics like programming languages, libraries, and databases were part of the discussions, many interviewees also called out heterogeneity topics beyond pure technology (TT14-TT17). Quality criteria of microservices' runtime, the internal architecture and code structure, and the coding style can also vary from microservice to microservice. The topic of heterogeneity in microservice-based projects is more extensive than we initially anticipated. For example, the development process models of microservice teams might be heterogeneous or organized in an overarching process framework (e.g., SAFe[5]).

This journey from focusing on purely technical topics to organizational management topics is also reflected by experience reports from the industry, such as by Spotify.[6] They accumulated many different technologies in their applications, some for similar use cases. To maintain an overview, they started to develop their homegrown developer portal *Backstage* that they open-sourced in 2020. It serves as an access point to their complete variety of infrastructure tooling in a unified (user) interface, as an onboarding tool, as a centralized place for technical documentation, and provides a service template functionality.

Our experience and industry projects, such as Backstage, are encouraging to conduct further empirical studies on the organizational success factors of microservices and their integration. Academic literature has only touched on such aspects until now, making this study one of the first to fully devote itself to a purely organizational perspective on successfully managing microservice-based projects in practice. Such organizational insights can, in turn, lead to new technical innovations and open-source projects like Backstage.

Throughout the interviews, we observed that most practitioners prefer a stricter standardization rather than leaving total technological freedom to the microservice teams. This conflicts with the often-cited benefit that software projects gain from using microservices. While this insight might not be representative of every microservice project and does not account for a temporal shift in general practitioner perception of heterogeneity in microservice projects, this is still a surprising outcome of this study. We attribute this perception to the increasing maturity of the microservice-based architectural style that seems to have passed the state of being a hype topic. The challenges and drawbacks of microservices have gained more awareness, leading to a healthy consideration of whether all aspects, like the complete independence of microservice teams, should always be implemented in real-life projects to their full extent.

6 Limitations

We use the trustworthiness criteria proposed by Guba (1981) to discuss the limitations of this study due to the qualitative nature of the empirically collected data. The following subsections will discuss the credibility, transferability, dependability, and confirmability of this study.

[5] https://scaledagileframework.com/

[6] https://engineering.atspotify.com/2020/03/what-the-heck-is-backstage-anyway

6.1 Limits to Credibility

The credibility criterion assesses the extent to which the findings accurately reflect the reality being studied.

Firstly, there is a potential influence of interviewer behavior and tone on interviewee responses. To mitigate this, we designed an interview guide in advance to ensure consistency and impartiality in questioning (Kallio et al. 2016). Additionally, we aimed to minimize the impact of individual researcher bias by having two interviewers conduct separate sets of interviews. The interviewers have experience in both academic research and industry practices within software engineering.

Secondly, the possibility of misunderstandings and misinterpretations during interviews and transcription could influence the accuracy of the results. To address this concern, we engaged in prolonged engagement with the data to mitigate careless errors Guba (1981). Moreover, we conducted member checking for the second set of interviews, allowing interviewees to review and provide feedback on the extracted insights, thereby enhancing the validity of our interpretations Guba (1981). In total, we received three answers that all confirmed our findings within a deadline of 3 weeks.

Thirdly, transparency in data analysis is crucial for establishing the believability of the results. While we cannot share interview transcripts due to confidentiality agreements, we have provided extensive supplementary materials summarizing how each interview contributes to the emergence of thematic patterns. Utilizing detailed descriptions and representative citations, we strive to convey a vivid and credible portrayal of each theme (Braun and Clarke 2006).

Lastly, while our study did not explicitly evaluate findings within an industry context, we recognize the importance of validating theoretical insights through practical application. Future work will involve applying the developed theory in industry settings to gain further insights into its nuances and contextual applicability. By bridging the gap between theory and practice, we aim to enhance the robustness and credibility of our findings.

6.2 Limits to Transferability

The transferability criterion assesses the extent to which the findings can be generalized beyond the specific context of the study.

Firstly, the sampling of interview participants may have introduced bias, potentially limiting the representation of various contexts necessary for building a generalizable theory. To mitigate this issue, we employed a sampling model with the goal to "[...] represent the diversity of the phenomenon under study within the target population" (Jansen et al. 2010), including participants with different expert roles, project sizes, and phases. Nonetheless, flaws in the sampling model could introduce selection bias. Therefore, we sought guidance from an expert in the field to identify hidden characteristics that may influence the facilitation of microservice-based projects (Jansen et al. 2010).

Secondly, the sample may not be exhaustive, not fully capturing all details on the topic. Conducting additional interviews could potentially yield deeper insights into the phenomenon under investigation. To address this concern, we tracked changes to the coding system throughout the analysis of each interview. We ceased further interviews once saturation was reached, indicating that additional interviews were unlikely to yield new relevant insights into the theory (Jansen et al. 2010). According to Hennink and Kaiser (2022), our sample size of 15 interviews is in the typical range of 9 to 17 interviews to reach saturation.

Thirdly, we acknowledge that the sample size is not large enough to make statistical generalizations. Accordingly, we make no use of statistical methods or quantitative claims about the underlying population. However, by achieving theoretical saturation and employing purposeful sampling, the findings provide theoretically rich insights that are transferable to similar contexts. This approach is consistent with qualitative research standards, emphasizing the contextual relevance and depth of understanding over numerical representativeness.

Lastly, temporal factors may diminish the transferability of results over time. The findings reflect the state of practice at the time of the interviews, which may become less relevant or applicable as circumstances evolve. Nevertheless, we posit that our theory is relatively insulated from the rapid pace of technological progress, as the primary driver of the underlying challenge lies in socio-technical processes, which are known to be less volatile.

6.3 Limits to Dependability

The dependability criterion assesses the extent to which the research design and execution is comprehensible and replicable.

Firstly, individual researchers may have an impact on data collection, particularly during the interview procedure. To mitigate this influence, we developed a comprehensive interview guide to standardize the structure of interviews and core questions (Kallio et al. 2016). Furthermore, to enhance methodological rigor, we implemented investigator triangulation (Thurmond 2001), with two interviewers conducting separate sets of interviews to minimize individual bias.

Secondly, thematic analysis relies on the researcher’s knowledge and perspective, potentially making data analysis less replicable. To address this concern, we conducted intercoder reliability sessions to ensure consistent interpretation of interview excerpts across multiple researchers (O’Connor and Joffe 2020).

Thirdly, while confidentiality agreements prevent us from sharing interview transcripts as supplementary materials, we have taken steps to ensure transparency and comprehensibility in our study. Our supplementary materials include representative quotes and summaries highlighting each interview’s contribution to thematic patterns (Braun and Clarke 2006). By providing these details, we aim to facilitate a deeper understanding of our research process and enhance the trustworthiness of our findings.

6.4 Limits to Confirmability

The confirmability criterion assesses the extent to which the biases and perspectives of the researcher shaped the results.

The interview guides served as a protocol to prevent derailing the interviews from their focus and asking questions in a reproducible manner. However, their design might have introduced a bias towards certain features of the phenomenon. To mitigate such a bias, we led with open questions first to get an unaffected viewpoint on a topic. Afterward, we occasionally asked for opinions on topics other interviewees brought up so we could get a better picture of certain features of the phenomenon. This approach manifests itself in Interview Guide 2 (see Appendix A.3), where we first generally asked about the advantages and disadvantages of technological heterogeneity before detailing its relation to cost and innovation speed. These two features, cost and innovation speed, arose from the first set of interviews as potentially interesting influencing trade-off factors. Even though this interview guide design might have influenced the data collection, we paid attention to not forcing any results during data analysis and reporting this potential bias. In this specific case, we were not able to find a common theme in the interviews for how innovation speed affects technological heterogeneity and vice versa. As the analysis was inconclusive on this aspect, we also didn't report any relationship. For costs, we were able to find such a relationship, even though it was a more complex one than we anticipated after the first set of interviews.

The data analysis procedure is subject to limitations as well. Firstly, thematic analysis inherently relies on the researcher's knowledge and perspective, which may introduce bias in the interpretation of data. Secondly, different researchers may interpret the same data differently, leading to potential variations in findings. Thirdly, unconscious selective reporting may favor particular themes or viewpoints, affecting the overall integrity of the study.

To mitigate these biases, we implemented several measures. To mitigate introducing biases, we complemented the continuous professional exchange between all co-authors and further members of our research group with regular peer debriefings to "[...] confirming that the findings and the interpretations are worthy, honest, and believable" (Spall 1998).

Additionally, inter-coder reliability sessions were conducted, necessitating the maintenance of a detailed codebook. The codebook documents the rationale behind each theme and code, along with criteria for their application (MacQueen et al. 1998). By formulating these criteria and soliciting qualitative feedback from the inter-coders, we strengthened the confirmability of our findings.

7 Conclusion

This article presents a study of technological heterogeneity (TH) in microservice-based projects and a theory of how to effectively balance the resulting trade-offs in alignment with project objectives.

Through interviews with industry practitioners, we developed a prescriptive multi-tier influence research model showcasing the factors influencing governance decisions regarding TH. We conceptualized TH as an n-dimensional spectrum, where each dimension represents a technological aspect, ranging from uniform to diversified choices across microservices. The delta of the current degree of TH to the optimal one, which is affected by the trade-off effects and the environmental factors of the project, is the main driver for governance in this regard. Further, we distilled 13 industry best practices that can guide the intent to change the current degree of TH towards the optimal one. The core best practice is introducing meaningful standardization as a governance tool to limit TH to a desirable degree, allowing essential but avoiding accidental heterogeneity. Standardizations should be deliberately created (e.g., by inter-team communities), knowledge about them should be actively dispersed (e.g., by documentation of standardization), and their application should be incentivized (e.g., by offering service templates) or enforced (e.g., by audits).

Our findings represent a pioneering effort to address the challenges of TH in microservice-based projects. While some of the results confirm ”common industry knowledge“, like using more technologies to increase operation and maintenance costs, they have never been empirically derived in this depth, and there is value in systematically confirming such assumptions (Tichy 2000). However, we also present new insights, like describing the governance process of balancing TH in a prescriptive multi-tier influence research model and compiling a set of best practices to implement such a governance. By shedding light on this underexplored area, our research serves as a catalyst for further theoretical development and empirical validation in academia. Practitioners can leverage our insights as actionable guidance to inform their technology governance practices. Notably, our adoption of the practitioner-focused pattern presentation format enhances our findings' applicability in real-world contexts.

While our research approach ensured robustness through data-driven analysis, it is imperative to acknowledge that they remain untested hypotheses. The gained insights require future statistical validation incorporating diverse industry settings to proof the generalizability of our findings. Future endeavors should aim to expand upon our work by examining TH dynamics in varied organizational contexts, thus enriching the theoretical framework and strengthening assertions regarding applicability. The hypotheses we presented in the discussion section may serve as a starting point for future work. Field experiments and online survey studies can add clarity to the topic by validating such hypotheses. We explicitly welcome other research groups building on our results to advance the field.

We generally advocate for an intensified focus on organizational topics in microservice-based projects. Even though microservices are ”just“ an architectural style, our interviewees gave us strong indicators that ”just“ considering it as an architectural choice isn't enough to lead such projects to success:

> *"[...] decisions on the management-political and organizational level, change and culture, and so on [...] are probably - actually quite certainly - the most challenging part if you want to [adopt microservices] the right way because you end up rethinking your whole IT [...]."* - Interview G4, translated from German

Future work might pick up the organizational best practices we found in the context of technological heterogeneity and transfer them to other organizational challenges. For example, teams for cross-cutting concerns or taskforce teams might be significant contributors to solving other socio-technical challenges in microservice-based projects. Exploring and inter-connecting organizational theories in this area has the potential further to ease the adoption of microservice architectures in practice.

In closing, it is clear that microservice-based projects are difficult to get right. As research continues to evolve on the topic, it is essential to remember that the people implementing software architectures and their interactions decide the success of projects and not a single technology.

Appendix

Best Practices

BP1: Meaningful standardization

Context:

There are multiple cross-functional teams, each responsible for one or more microservices.

Problem:

(1) The diversity of technology (tech stacks, programming languages, frameworks, data stores) leads to additional costs, and limits the effectiveness of moving people from one team to another due to high knowledge barriers.

(2) The stringent enforcement of one specific technology stack limits the flexibility and introduces complexity by using not suiting tools.

Solution:

Introduce meaningful standardization to welcome essential technological heterogeneity and prevent accidental one. Establish processes and mechanisms to (i) create standardization, (ii) disperse the knowledge of their existence and how to implement them across the microservices teams, and (iii) consistently apply them.

Sources of evidence:

T1, T2, T3, T4, T5, T6, T7, T8, T9, G1, G2, G4, G5, G6

Representative quotes:

"It makes sense to define a set of minimal standards so that [the microservice teams] don't have to start at zero when coordinating, so that coordination can be reduced to the minimal amount that is necessary. It is a trade-off at the end of the day. [...] The value curve is kind of a bell-shaped curve or a parabola, logically speaking. It has a sweet spot. Too little is not good; too much isn't good either." - G4, translated from German

BP2: Inter-team communities

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1).

Problem:

Global decisions for standardization lack the backing of the teams, are questioned, or are even partially ignored.

Solution:

Establish inter-team communities for decision-making, like standardization. Communities revolve around certain subjects, usually around a cross-cutting concern or a technology type. Subject-savvy and interested members of the different microservice teams can participate in the communities and represent their team in the community. The community (a) keeps an overview of their topic, (b) makes critical decisions, (c) assigns the responsibility of shared capabilities or bears it itself, and (d) disperses information across teams by their representatives.

Sources of evidence:

T1, T2, T4, T5, T6, T7, T8, T9, G2, G3, G4, G5, G6

Representative quotes:

"We like to speak of micro and macro architecture or micro and macro decisions. Micro is everything I can decide on my own. Macro would be something like the API design guide [...], or a security concept, or which tracing infrastructure and format to support. [...] these are macro decisions that someone has to make. Usually a group. What we frequently do is that every service team sends an architecture-savvy representative to an architecture board. And there exactly these decisions are made [...]." - G2, translated from German

BP3: Teams for cross-cutting concerns

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1).

Problem:

Global decisions for standardization of cross-cutting concerns lack the backing of the teams since it puts more load on the microservice teams to adhere to the standardizations.

Solution:

Establish a team responsible for the cross-cutting concern as an organizational unit. This team makes decisions on standardization on the cross-cutting concern and supports the microservice teams adhering to the standardization. For example, an operations team could operate and maintain certain standardized databases and provide monitoring capabilities to the microservice teams that adhere to the standardization.

Sources of evidence:

T1, T2, T3, T5, T7, T8, T9, G1, G2, G5, G6

Representative quotes:

"Most often, there is a dedicated team for the things in technology in the mainline that is responsible. For us, it is a monitoring team, there is also a team doing the databases." - T7, translated from German

BP4: Taskforce team

Context:

The project is in the beginning stages of adopting a microservice-based architecture. There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1).

Problem:

Microservice teams cannot live up to the knowledge needs required for microservices and the introduced standardizations.

Solution:

Introduce a task force team that acts as a front-running team. It builds up the required knowledge about technologies, makes decisions on technology standardization, and supports the microservice teams to catch up and adhere to the standardizations.

Sources of evidence:

T1, T2, T4, T6, G1

Representative quotes:

"So we do have a formal core services team. And that happened because we needed, essentially, we needed to like really buckle down on this microservices approach and we needed to build out a lot of this stuff very quickly. So so we ended up establishing this team. So we had somebody with like the sole responsibility for it." - T4

BP5: Deliberate experimentation

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1).

Problem:

The current standardization does not satisfy all use cases efficiently.

Solution:

Deliberately plan and allocate resources for experimentation phases. Proof of concepts and prototypes can be utilized to evaluate the fit of a new technology and whether it is worth the cost of introducing and maintaining an additional technology. Experimentation might mean spending effort without a positive outcome. These expenses should be weighed against the benefits an additional technology may introduce.

Sources of evidence:

T1, T3, T4, T5, T6, T7, T8, T9, G1, G2, G4

BP5: Deliberate experimentation

Representative quotes:

"And it was actually a common thing to start when a new feature was to be developed to start with a little POC and evaluate what would be the best technology for to address that. Which database is the best fit, for example." - T9

BP6: Documentation of standardization

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1). A standardization has been created or updated.

Problem:

The information of standardization decisions does not flow to the microservice teams which should adhere to the standardization.

Solution:

Document the standardization at a defined place. Documenting not only the outcome but also the considerations surrounding technology decisions serves as a look-up for the microservice teams. Developers can inspect which technology is usable without any discussion. Organizing all standardization documentation in one location instead of dispersing it to multiple locations improves discoverability.

Sources of evidence:

T1, T2, T4, T5, T9, G2, G5

Representative quotes:

"But this was documented then in the end of the session in the Confluence [wiki] page. And there was also a complete tech stack page that every engineer could look up and just see what were green-lighted techs that they could use already without discussion." - T5

BP7: Regular cross-team discussions

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1). A standardization has been created or updated.

Problem:

Situations arise that point to a lack of knowledge transfer between microservice teams. The information of standardization decisions does not flow to the microservice teams which should adhere to the standardization.

Solution:

Organize a regular cross-team discussion meeting serving to convey knowledge between teams. It can be as simple as reserving a fixed time slot every week and offering a stage for presenting progress, insights, and established standardization that might be relevant for all teams. Teams can send representatives that will take the knowledge into their teams.

Sources of evidence:

T3, T4, T9, G4, G6

Representative quotes:

"There are environments where the teams are allowed to introduce whatever they want to. [...] the only enforcement is they have to speak about it. So whatever they've learned, they have to share it with the other teams and they have to give a town hall session about new technology they have tested or introduced and stuff like this. And from my perspective, even this is not an enforcement, but an incentive." - T3

BP8: Education programs

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1). A standardization has been created or updated.

BP8: Education programs

Problem:

Situations arise that point to a lack of knowledge to implement some standardization. The information on how to adhere to some standardization decisions does not flow to the microservice teams.

Solution:

Establish an education program for the microservices teams. Education programs can support overcoming the knowledge hurdles to do microservices and adhering to the created standardization in an efficient way. Education programs can come in different formats, like lectures, workshops, or blueprints and tutorials on how to adhere to a standardization.

Sources of evidence:

T1, T6, T7, G1

Representative quotes:

"Yeah, but it would definitely help if you would have some standard decisions and also maybe some templates and some blueprints. For instance, like the upgrade from .NET 3 to 6 - if we stay with that example - it's actually quite easy. If you know what to do, you need to replace this and that. And usually, everything is fine afterward. But it would help a lot if you have that standard and say: Please teams take a look here. That's how you do it. It would not cost you more than one day. Please do it. If you have any questions, come to us." - T1

BP9: Technology sign-off process

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (P1). A standardization has been created or updated. The microservice teams are aware of that and have the knowledge to adhere to the standardization

Problem:

Some microservice teams do not adhere to the standardization.

Solution:

Introduce a mandatory technology sign-off process. Every time a microservice team wants to use a previously unused technology, they have to get the technology signed off. This process empowers an organizational entity to enforce the tech stack and only introduce essential and no accidental technological heterogeneity.

Sources of evidence:

T3, T5, T7, T9

Representative quotes:

"But later on, it went all through the body of this tech council, tech committee. That means when a team wanted to add new software or even nothing about adding new software, but just starting a new project, there was at least a short discussion with the tech council to just see and discuss what's necessary. What kind of databases do they need? What kind of infrastructure support do they need? And then also discussion, what kind of tech are they going to use? And reuse maybe even existing tech that we had in the company." - T5

BP10: Audits for the detection of deviations

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (BP1). A standardization has been created or updated. The microservice teams are aware of that and have the knowledge to adhere to the standardization

Problem:

Some microservice teams do not adhere to the standardization.

Solution:

Introduce a regular auditing process to check and enforce that microservice teams adhere to the standardization. Deviations can be detected and discussed. Audits can take the shape of parts of a formal audit in domains that are regulated anyway, e.g., the health sector, or be part of informal audits, e.g., code reviews.

BP10: Audits for the detection of deviations

Sources of evidence:

T2, T5, T6, T9, G1, G6

Representative quotes:

"There were periodic audits where certain things had to be there. Not everything from the beginning, but there were various milestones that defined that some things needed to be implemented when reaching them in our project regarding procedures, processes, and standards" - G6, translated from German

BP11: Automation of the deviation detection

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (BP1). A standardization has been created or updated. The microservice teams are aware of that and have the knowledge to adhere to the standardization

Problem:

Some microservice teams do not adhere to the standardization.

Solution:

Automate the detection of deviations from the standardization, e.g., as software tests in the CI pipeline. Making the standardization measurable and bringing it to the attention of the developers enforces or incentivizes adherence to the standardization, depending on whether the CI pipeline fails or just logs a warning. If furnished with a clue on how to implement the standardization, such automatization can also serve as a communication tool to disperse knowledge.

Sources of evidence:

T1, T2, T4, T9, G1, G2, G6

Representative quotes:

"Similar to if you use Sonar cloud for checking your code quality. [...] So having like a measurable standard. All automated for sure. That would help a lot. That you have this one dashboard where it says, okay, these two services are running behind the standard. [...] So I would not enforce but rather kind of point out that the standard is violated. With a clear indication on how to fix it." - T1

BP12: Service templates

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (BP1). A standardization has been created or updated. The microservice teams are aware of that and have the knowledge to adhere to the standardization

Problem:

Some microservice teams do not adhere to the standardization.

Solution:

Establish and maintain a template for a quicker ramp-up of new microservices. Such a starting point for new microservices incentivizes sticking to the standardization by the convenience of software reuse. Using different technology would require efforts to develop an equivalent starting point for a new microservice.

Sources of evidence:

T1, T4

Representative quotes:

"So basically the standards were set by the two teams. But following that, it's a bit fuzzy [...] we do have a template for microservices, which is also more or less up to date." - T1

BP13: Collaboration on cross-cutting tools and libraries

Context:

There are multiple cross-functional teams, each responsible for one or more microservices. The project aims to introduce meaningful standardization (BP1). A standardization has been created or updated. The microservice teams are aware of that and have the knowledge to adhere to the standardization

BP13: Collaboration on cross-cutting tools and libraries
Problem:
Some microservice teams do not adhere to the standardization.
Solution:
Introduce an inner-source or open-source process to foster collaboration between teams on cross-cutting tools and libraries. Reusing such tools and libraries incentivizes sticking to the standardization by the convenience of software reuse. Using different technology would require efforts to re-implement an equivalent solution. These shared projects should target reusable general-purpose code and not domain-specific aspects like shared models to avoid introducing coupling between microservices.
Sources of evidence:
T1, T2, T6, T7, T8, T9, G5
Representative quotes:
It's as I mentioned previously, we have lots of libraries in place that already force you into a certain direction. [...] And if you use the libraries we provide, it's pretty much standardized then again. But you would have the freedom to do something else. But it's more convenient just to use the libraries currently, at least. - T2

Qualitative Survey - Interview Guide 1 (Microservice Integration)

Interview Guide 1 aimed to explore the challenges and existing solutions in microservice integration. Accordingly, the questions were open-ended and broad in scope. While technological heterogeneity was not explicitly addressed, it emerged organically during the interviews. Using probing techniques, we posed follow-up questions where appropriate. These insights motivated a more focused follow-up study on technological heterogeneity (see Appendix A.3). Additionally, we used these interviews as the foundation for our thematic analysis.

Phase 1: Preamble

- Small talk
- Introduce us and our research context
- Information about confidentiality and data handling
- Which language is preferred? English vs. German?
- Audio recording ok?

Phase 2: Warm-up Questions

- Please present yourself, your role and responsibilities at your company.
- Since when are you using microservices?
- What is a microservice for you?
- Why should you use microservices? Why should you not?

Phase 3: Definition Microservice Integrationw

- What is microservice integration to you?

Phase 4: Microservice Integration Techniques

For each topic identified in phase 3:

- Why is this topic important?

- What are the goals in that topic?
- How do/did you achieve these goals?
- Which problems are/were preventing you from achieving these goals?
- What are solutions you found to these problems?
- What else is important for that topic?

Phase 5: Cool-down

- What aspect was not mentioned in this interview that you would like to be part of our research?
- Can you recommend someone equally knowledgeable in microservices that we could contact for our research?

Qualitative Survey - Interview Guide 2 (Technological Heterogeneity)

Interview Guide 2 aimed to deepen our understanding of balancing technological heterogeneity in microservice-based projects. Accordingly, the questions focused exclusively on this topic. We also employed probing techniques to clarify misunderstandings and reveal implicit knowledge. Following the interviews, we extended the initial code system by incrementally coding this new data.

Phase 1: Preamble

- Small talk
- Introduce us and our research context
- Information about confidentiality and data handling
- Which language is preferred? English vs. German?
- Audio recording ok?

Phase 2: Warm-up Questions

- Please present yourself, your role, and your responsibilities at your company.

Phase 3: Technological Heterogeneity in General

- How heterogeneous are the technologies in your microservice project(s)?
- What are the advantages and disadvantages of technological heterogeneity in your experience?
- How is technological heterogeneity related to the innovation speed of the project?
- How is technological heterogeneity related to the costs of the project?

Phase 4: Governing Technological Heterogeneity

- How do you manage the heterogeneity of technologies in your microservice project(s)?
- Which kinds of technologies are managed? Is there a rule of thumb on what to manage and what not?

– Who is responsible for what in the government process?
– How is uniformity enforced or incentivized?

Phase 5: Cool-down

– Which aspects of the topic did we not yet discuss but are especially important to you?
– Can you recommend someone equally knowledgeable in microservices that we could contact for our research?

Acknowledgements The present work was performed in partial fulfillment of the requirements for a cumulative dissertation. We thank our colleagues for their continuous feedback and proofreading of this article. Finally, we are very grateful for the support we received from the industry, especially from the interviewees who donated their valuable time to support our research, such as Aron Metzig, Benjamin Klatt (viadee), Jonas Schüll, Rasmus Roy Tillgaard, Sebastian Denel, Sebastian Goeb (WS Audiology), and Stefan Appel.

Author Contributions Georg-Daniel Schwarz: Conceptualization (equal), Data curation (lead), Formal analysis (lead), Funding acquisition (supporting), Investigation (equal), Methodology (equal), Project Management (lead), Software (lead), Validation (equal), Writing - original draft (lead), Writing - review and editing (equal); Philip Heltweg: Conceptualization (supporting), Data curation (supporting), Formal analysis (supporting), Investigation (equal), Methodology (equal), Project Management (supporting), Validation (equal), Writing - review and editing (equal); Dirk Riehle: Conceptualization (equal), Funding acquisition (lead), Investigation (supporting), Methodology (equal), Resources (lead), Supervision (lead), Writing - review and editing (equal).

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by BMBF (Federal Ministry of Education and Research) Software Campus 2.0 project (BePra-MSI, 01IS17045), and by DFG (German Research Foundation) Research Grants Programme (Industry Best Practices for Microservice Integration, RI 2147/9-1).

Data Availability Supplementary materials were are available on Zenodo: https://doi.org/10.5281/zenodo.15632039. Due to confidentiality agreements, we cannot make the interview recordings or transcripts publicly available. However, the supplementary materials summarize how the interviews contributed to the individual elements of our theory by summarizing their content to the specific points.

Declarations

Ethical Approval The study involved non-sensitive interview data and did not pose any risks to the participants. We did not seek approval by an ethics board because it is neither required nor standard in this kind of study at our institution.

Informed Consent Informed verbal consent was obtained from all participants before the commencement of the interviews. Participants were informed about the study's aims, their right to confidentiality, and their freedom to withdraw at any time without consequence.

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

Clinical Trial Number Not applicable.

References

Balalaie A, Heydarnoori A, Jamshidi P, Tamburri DA, Lynn T (2018) Microservices migration patterns. Softw Pract Exp 48(11):2019–2042

Baškarada S, Nguyen V, Koronios A (2018) Architecting microservices: practical opportunities and challenges. J Comput Inf Syst

Bogner J, Zimmermann A (2016) Towards integrating microservices with adaptable enterprise architecture. In: 2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW), pp 1–6

Braun V, Clarke V (2006) Using thematic analysis in psychology. Qual Res Psychol 3(2):77–101

Brooks FP, Bullet NS (1987) Essence and accidents of software engineering. IEEE Comput 20(4):10–19

Campbell JL, Quincy C, Osserman J, Pedersen OK (2013) Coding in-depth semistructured interviews: problems of unitization and intercoder reliability and agreement. Sociol Methods Res 42(3):294–320

Chen L (2018) Microservices: architecting for continuous delivery and DevOps. In: 2018 IEEE International conference on software architecture (ICSA). IEEE, pp 39–397

Clarke V, Braun V, Hayfield N (2015) Thematic analysis. Qualitative psychology: a practical guide to research methods, vol 222, pp 248

Di Francesco P, Lago P, Malavolta I (2018) Migrating towards microservice architectures: an industrial survey. In: 2018 IEEE International Conference on Software Architecture (ICSA). IEEE, pp 29–2909

Eisenhardt KM, Graebner ME (2007) Theory building from cases: opportunities and challenges. Acad Manag J 50(1):25–32

Guba EG (1981) Criteria for assessing the trustworthiness of naturalistic inquiries. Ectj 29(2):75–91

Guion LA, Diehl DC, McDonald D (2011) Triangulation: establishing the validity of qualitative studies: FCS6014/FY394, Rev. 8/2011. Edis 8:3–3

Harms H, Rogowski C, Iacono LL (2017) Guidelines for adopting frontend architectures and patterns in microservices-based systems. In: Proceedings of the 2017 11th joint meeting on foundations of software engineering, pp 902–907

Hennink M, Kaiser BN (2022) Sample sizes for saturation in qualitative research: a systematic review of empirical tests. Soc Sci Med 292(2022):114523

Jamshidi P, Pahl C, Mendonça NC, Lewis J, Tilkov S (2018) Microservices: the journey so far and challenges ahead. IEEE Softw 35(3):24–35

Jansen H et al (2010) The logic of qualitative survey research and its position in the field of social research methods. In: Forum qualitative sozialforschung/forum: qualitative social research, vol 11

Kallio H, Pietilä A-M, Johnson M, Kangasniemi M (2016) Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. J Adv Nurs 72(12):2954–2965

Krylovskiy A, Jahn M, Patti E (2015) Designing a smart city internet of things platform with microservice architecture. In: 2015 3rd international conference on future internet of things and cloud. IEEE, pp 25–30

Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 159–174

MacQueen KM, McLellan E, Kay K, Milstein B (1998) Codebook development for team-based qualitative analysis. Cam J 10(2):31–36

Márquez G, Astudillo H (2018) Actual use of architectural patterns in microservices-based open source projects. In: 2018 25th Asia-Pacific Software Engineering Conference (APSEC). IEEE, pp 31–40

Motulsky SL (2021) Is member checking the gold standard of quality in qualitative research? Qual Psychol 8(3):389

Myers MD, Newman M (2007) The qualitative interview in IS research: examining the craft. Inf Organ 17(1):2–26

Newman S (2021) Building microservices. O'Reilly Media"

O'Connor C, Joffe H (2020) Intercoder reliability in qualitative research: debates and practical guidelines. Int J Qual Methods 19(2020):1609406919899220

Palvia P, Midha V, Pinjani P (2006) Research models in information systems. Commun Assoc Inf Syst 17(1):47

Rademacher F, Sachweh S, Zündorf A (2019) Aspect-oriented modeling of technology heterogeneity in microservice architecture. In: 2019 IEEE International conference on software architecture (ICSA). IEEE, pp 21–30

Rademacher F, Sorgalla J, Wizenty P, Sachweh S, Zündorf A (2020) Graphical and textual model-driven microservice development. Microservices: science and engineering, pp 147–179

Riehle D, Züllighoven H (1996) Understanding and using patterns in software development. Tapos 2(1):3–13

Schwarz G-D, Bauer A, Riehle D, Harutyunyan N (2025) A taxonomy of microservice integration techniques. Inf Softw Technol 2025:107723

Spall S (1998) Peer debriefing in qualitative research: emerging operational models. Qual Inq 4(2):280–292

Strauss A, Corbin J (1998) Basics of qualitative research techniques

Taibi D, Lenarduzzi V, Pahl C (2018) Architectural patterns for microservices: a systematic mapping study. In: Proceedings of the 8th international conference on cloud computing and services science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018, Muñoz VM, Ferguson D, Helfert M, Pahl C (eds). SciTePress, pp 221–232

Thurmond VA (2001) The point of triangulation. J Nurs Scholarsh 33(3):253–258

Tichy WF (2000) Hints for reviewing empirical work in software engineering. Empir Softw Eng 5(4):309–312

Weerasinghe S, Perera I (2022) Taxonomical classification and systematic review on microservices. Int J Eng Trends Technol 70(3):222–233

Widjaja T, Gregory RW (2012) Design Principles for heterogeneity decisions in enterprise architecture management