A Heterogeneous Approach to Agile Tailoring

Abdallah M. Salameh

UNIVERSITY OF SALFORD

SCHOOL OF COMPUTING, SCIENCE & ENGINEERING

AUGUST 19, 2020

This document is toward the degree of Doctor of Philosophy

Abstract

Self-organisation is recognised as one of the fundamental principles of Agile software development since the introduction of the Agile Manifesto. In large-scale agile, self-organising teams must cooperate to produce a software product jointly. Previous research has identified barriers to self-organising autonomous teams in large-scale agile, such as lack of guidelines for how teams should be organised, and other challenges related to building autonomous teams and aligning them. The Spotify model is an example of a very large-scale agile approach, which is driven by creating loosely coupled, yet tightly aligned squads. However, there is a conflicting trade-off between squads autonomy and alignment. Too much alignment might hinder squad autonomy, but without alignment, the squads are autonomous but are not effective.

This research aims to (1) explore how Agile practitioners, from a FinTech organisation, resolve the conflicting trade-offs between squads autonomy and alignment, and (2) develop and evaluate new architectural governance practices, in a FinTech organisation tailoring the Spotify model. To address these aims, a qualitative research design was utilised. This research comprises two parts, a longitudinal embedded case study and an intervention embedded case study, which were conducted in a FinTech organisation (i.e., FinTechOrg).

The longitudinal embedded case study lasted over 21 months, during which 225 ceremonies were observed, 14 semi-structured open-ended interviews were conducted, and data were collected from different sorts of artefacts. The collected data were analysed using an approach informed by the Grounded Theory method.

The analysis identified influential factors on different aspects of Spotify Tailoring, which contribute to resolving the conflicting trade-offs between squads autonomy and alignment. These aspects are (1) establishing and building autonomous squads, (2) aligning autonomous squads, and (3) performing B2B product development by tailoring the Spotify model. The first contribution is identifying factors influencing establishing and building autonomous squads. The second contribution is identifying factors influencing aligning autonomous squads. The third contribution is identifying the impact of product

development on Spotify tailoring as well as factors influencing Spotify Tailoring for B2B product development. Each identified factor is supported by a set of practices and attributes, which can aid Agile practitioners in improving squad autonomy, aligning autonomous squads, or facilitating the conduct of B2B product development.

The fourth contribution is identifying a novel approach to Agile tailoring, called *Heterogeneous Tailoring*. Three key features characterise this approach. Firstly, each autonomous squad is empowered to select and tailor its development method. Secondly, each squad is aligned with other squads and to common product development goals and objectives. Thirdly, the product steering committee draws the strategy of squads' missions and aligns the product backlog among autonomous squads. This novel approach to Agile tailoring has improved the creativity for some squads and increased the productivity for others, as reported by the practitioners of FinTechOrg.

The Spotify model and the revealed Heterogeneous Tailoring approach do not provide practices or guidelines for governing Agile architecture across autonomous squads. Thus, an embedded case study intervention was conducted to contribute to facilitating Agile architecture governance. In this intervention, a novel approach to architectural governance was developed and evaluated in FinTechOrg, which is considered the last contribution. This approach incorporates a structural change and an architecture change management process. The intervention lasted 3 months, during which 32 ceremonies were observed and 8 semi-structured open-ended interviews were conducted. The collected data was analysed using an approach informed by the Grounded Theory method.

Based on the results of the intervention, the Heterogeneous Tailoring approach was adapted to accommodate the novel approach to architectural governance. This adaptation has impacted the key features of the Heterogeneous Tailoring approach. Establishing autonomous squads was impacted by the introduced structural change. The alignment of autonomous squads was impacted by governing Agile architecture. Product development was impacted by the needs for planning architecture based user stories. Also, the adaptation of the Heterogeneous Tailoring approach revealed a new key feature, called Release Strategy, which is concerned with the continuous delivery of architecture enablers.

Acknowledgement

"O' Lord! Increase me in knowledge" – Holy Quran (20:114)

I am indebted to my supervisor, Dr Julian M. Bass, for supporting me through the ups and downs of this long journey, challenging me, and for drawing out the best in me. Without your continuous help and support, this research would not be possible.

My deepest gratitude goes to my parents for their endless love, support and continuous encouragement; my dear brothers, Iyad, Tariq, Habeeb, Ahmad, Hatem, and Mahmoud, for being always there when I need you; my lovely wife, Hind, for being my pillar of strength and best friend; my sons, Habeeb, Yones, Sulieman, for being my source of inspiration; and parents-in-law for the delicious food.

My sincere regard goes to all of those who supported me in any respect during this journey. I do not mention you all here, but I hold you in high esteem.

Contents

List of Tables

List of Figures

Terms and Definitions

- The Spotify Model: This is Spotify's agile approach, which was initially introduced by Kniberg and Ivarsson [58] and described in detail in [57, 56].
- Spotify Tailoring: The tailoring of the Spotify model.
- FinTechOrg: A FinTech organisation, which is a subject of this research study.
- FinTech's Spotify model: The Agile approach adapted in FinTechOrg, which has tailored the Spotify model, and explored in this research study.
- Large-scale agile: Agile software development efforts that involve a large number of actors, less than 100 members, and distributed over 2 up to 9 teams.
- Very Large-scale agile: Agile software development efforts that involve a large number of actors, more than 100 members, and distributed over more than 9 teams.
- Spotify communities: A Spotify community is an agile unit with a commonality such as norms, values, customs, or mission. Community members share a sense of place situated in a given organisation. A community also draws communication among its members. The Spotify model introduces multiple communities, namely: Squad, Chapter, Guild, and Tripe.
- Squad: An autonomous team that has an identified mission. A squad does not have a formally appointed Squad Leader but has a specific appointed Product Owner.
- Tripe: A collection of Squads that are designed to be smaller than 100 people and aims to minimise dependencies that can slow or obstruct a squad.
- Chapter: A group of people located within a Tripe, which have similar skills and working within the same competency area.
- Guild: A group of people who are wide-reaching within the organisation and have a desire to share knowledge, tools, code, and practice across the whole organisation. While Chapters are always located within a Tribe, a Guild contains members that are distributed across the whole organisation.

Chapter 1

Introduction

1.1 Introduction

Organisations usually tailor Agile methods to fit project-specific requirements as well as organisations' values, strategies, and culture [17, 24]. Bearing in mind that organisations have different cultures and contexts, even two projects within the same organisation can be different. What can be applied successfully for a specific project, can be inapplicable for another project, which has a different scope, domain, customers, etc., within the same organisation.

Two main approaches to Agile tailoring have been identified by previous research [17, 24, 35]: Contingency Factors and Method Engineering. The Method Engineering approach utilises meta-method processes to create new customised Agile methods [24]. These new methods are based on selected practices and processes from well-known Agile methods to be applied on a specific context. The Contingency Factors approach selects multiple Agile methods to be available to the organisation [35]. Then, an Agile method selection is performed based on project scope, domain, context, and culture.

The Spotify company, which was founded in Sweden in 2006, has a music streaming service. This service has benefited from substantial growth and had a total of 217

million monthly active users worldwide in April 2019 [88]. The Spotify company has developed its own Agile culture, by tailoring Lean and Scrum, to facilitate software development for hundreds of developers across 3 cities [58]. The Spotify model is an example of a very large-scale agile tailoring, which is driven by creating autonomous teams (i.e., squads) [57, 58]. This model employs numerous autonomous squads to promote sustainable creative development.

1.2 Research Problem and Motivation

Large-scale projects are challenging because several teams need to work together to release a single product [23, 26, 79]. Self-organisation does directly influence teams' effectiveness since the authority of decision making is moved to the operational level, which increases the speed and accuracy of problem-solving [76]. However, these autonomous teams should have common objectives and be able to organise themselves to overcome encountered challenges [48, 55]. Since self-organisation is recognised as a hallmark of Agile software development over the last decade, large-scale organisations started emphasising the necessity of employing autonomous teams to utilise effective development [50, 76]. The Spotify model is an example of large-scale Agile tailoring, which is driven by creating loosely coupled (i.e., autonomous), yet tightly aligned squads (i.e., teams) [56, 57, 58].

This research study was motivated by my identification of conflicting trade-offs between squads autonomy and alignment in the Spotify model. Hence, the research problem is presented in how agile practitioners can resolve and balance the conflicting trade-offs between squads autonomy and alignment. The Spotify model initiates the creation of autonomous yet aligned squads by utilising an adaptive organisational structure [57, 58]. However, this model does not provide guidelines for balancing squads autonomy and alignment. The Spotify model does not provide guidelines for: (1) establishing and building autonomous squads, (2) aligning autonomous squads, (3) performing product development in other contexts. Also, this research study identified a lack of research into the Agile tailoring approach used in the Spotify model. Moreover, there is a lack of

approaches to architectural governance using the Spotify model. Indeed, there is a lack of scientific research on the Spotify model across multiple projects, organisations, and cultures.

1.3 Research Aims, Objectives, and Questions

1.3.1 Aims

The aims of this research are to (1) explore how Agile practitioners, from a FinTech organisation, resolve the conflicting trade-offs between squads autonomy and alignment, and (2) develop and evaluate new architectural governance practices, in a FinTech organisation tailoring the Spotify model.

1.3.2 Objectives

To achieve these research aims, the following research objectives were developed:

- **Objective 1:** Describe approaches used, in a FinTech organisation, to achieve and sustain squad autonomy in practice.
- **Objective 2:** Classify the Agile tailoring approach used in the FinTech's Spotify model.
- **Objective 3:** Tailor the FinTech's Spotify model for architectural governance.

1.3.3 Questions

These research objectives were designed to answer the following research questions:

- **RQ1:** How do Agile practitioners, from a FinTech organisation, describe approaches to the balance between squad alignment and autonomy?
  - **RQ1.1:** How do Agile practitioners, from a FinTech organisation, establish and build squad autonomy?
  - **RQ1.2:** How do Agile practitioners, from a FinTech organisation, achieve and sustain the alignment of autonomous squads?
  - **RQ1.3:** How do Agile practitioners, from a FinTech organisation, tailor the Spotify model for B2B product development?
- **RQ2:** What is the approach taken to Agile tailoring, when using the Spotify model?

  RQ3: What new architectural governance practices can be introduced for loosely coupled yet cooperating squads?

1.4 Research Design

To fulfil the aims and objectives of this research study, a *qualitative design* was employed. This qualitative research design comprises two parts: a longitudinal embedded case study and an intervention embedded case study.

A longitudinal embedded case study was conducted to explore Spotify Tailoring in a multinational FinTech organisation. This longitudinal embedded case study lasted over 21 months, during which 225 ceremonies were observed, and 14 semi-structured open-ended interviews were conducted. The collected data were analysed using an approach informed by the Grounded Theory.

The analysis of this longitudinal embedded case study revealed a novel approach to Agile tailoring, called *Heterogeneous Tailoring*. Three key features characterise this approach. Firstly, each autonomous squad is empowered to select and tailor its development method. Secondly, each squad is aligned with other squads and to common product development

goals and objectives. Thirdly, product steering committee draws the strategy of squads' missions and aligns the product backlog among autonomous squads. However, two main challenges to the Heterogeneous Tailoring approach were identified by this research. Firstly, it is difficult to measure the overall code quality, performance, and productivity for different squads. Secondly, aligning and governing architectural decisions across autonomous squads requires new practices.

To overcome the second challenge, this research developed a novel approach to Agile architecture governance. Then, an intervention embedded case study was conducted to evaluate the developed approach. This intervention lasted over 3 months, during which 32 ceremonies were observed, and 8 semi-structured open-ended interviews were conducted. The collected data was analysed using an approach informed by the Grounded Theory method.

1.5 List of Publications

The following research papers were conducted throughout this research. Fig. 1.1 illustrates how the research questions correspond to the contributions mentioned above as well as the following peer-reviewed research papers.

- Spotify Tailoring for Promoting Effectiveness in Cross-Functional Autonomous Squads. In Agile Processes in Software Engineering and Extreme Programming - Autonomous Teams Workshop (XP, A-Teams). [94].
- Influential Factors on Aligning Spotify Squads in Mission-Critical and Offshore Projects - a Longitudinal Embedded Case Study. In Product-Focused Software Process Improvement (PROFES). **Received Best Full Research Paper Award**. [93].
- Spotify Tailoring for B2B Product Development. In Euromicro Conference on Software Engineering and Advanced Applications (SEAA). [95].
- Heterogeneous Tailoring Approach Using the Spotify Model. In Proceedings of the Evaluation and Assessment on Software Engineering (EASE). [98].

- Spotify Tailoring for Architectural Governance. In Agile Processes in Software Engineering and Extreme Programming - Autonomous Teams Workshop (XP, A-Teams). [97].
- An Architectural Governance Approach by Tailoring the Spotify Model. AI & Society: Journal of Knowledge, Culture and Communication. Special issue on team autonomy in digital transformations. [96].
- A Safety-centric Change Management Framework by Tailoring Agile and V-Model Processes. In International System Safety Conference (ISSC). [99].

1.6 Thesis Structure

This thesis consists of the following chapters:

Chapter 1 Introduction: Describes the research problem and motivations behind this research study, the research aims, objectives, and questions, the employed research design, the contributions of this study, and the structure of this thesis.

Chapter 2 Literature Review: Presents an overview of related literature. In keeping with the research design and employed methods (described in Chapter 3), a minor literature review was conducted upfront. Also, a detailed literature review is presented to position the research findings and discussion.

Chapter 3 Research Design: Presents the research approach (philosophical worldview, design, methods) employed in this study. This chapter presents the research methods used in data collection and analysis. Also, this chapter describes the conducted longitudinal and intervention embedded case studies.

Chapter 4 Spotify Tailoring for Establishing and Building Autonomous Squads: Identifies influential factors on establishing and building autonomous squads.

Chapter 5 Spotify Tailoring for Aligning Autonomous Squads: Identifies influential factors on aligning autonomous squads by investigating Spotify Tailoring.

Chapter 6 Spotify Tailoring for B2B Product Development: Identifies two main aspects related to product development using the Spotify model. Firstly, The impact of product development on squads autonomy and alignment was identified. Secondly, performing B2B product development by tailoring the Spotify model was identified.

Chapter 7 A Heterogeneous Tailoring Approach to Agile Tailoring: Reveals a novel approach to Agile tailoring (i.e., Heterogeneous Tailoring).

Chapter 8 Spotify Tailoring for Architectural Governance: Presents an evaluates an approach to architectural governance. Also, this chapter adapts the Heterogeneous Tailoring approach, which is presented in Chapter 7, to conform with the evaluated approach to architectural governance.

Chapter 9 Discussion: Analyses and interprets presented findings in Chapters 4-8 based on related literature, which is presented in Chapter 2. Also, this chapter analyses the limitations of the study.

Chapter 10 Conclusion: Describes and summarises the contributions of this thesis.

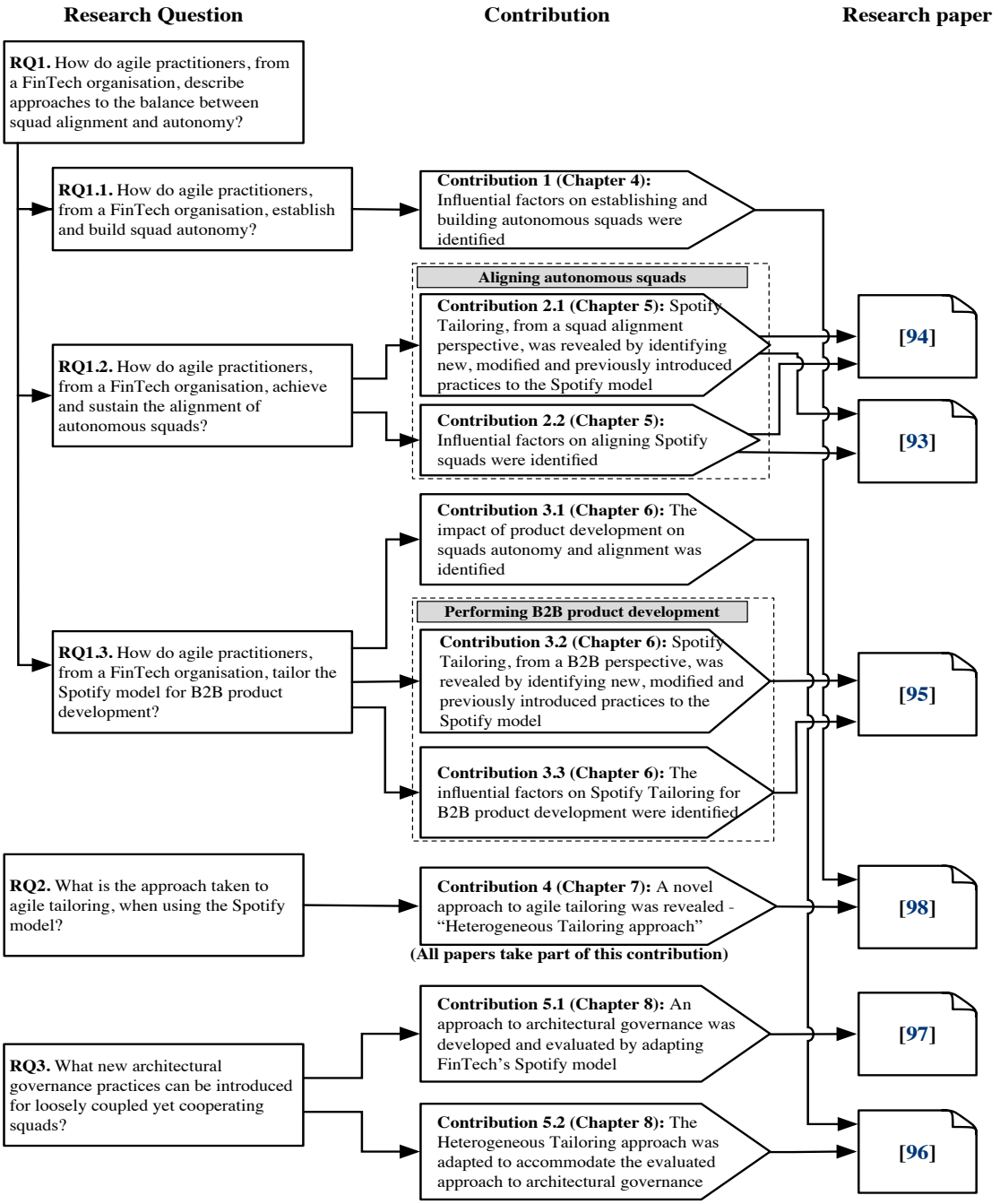


Figure 1.1: The connections between the research questions, contributions, and conducted research papers

Chapter 2

Literature review

2.1 Introduction

This chapter presents an overview of Agile software development and Agile method tailoring. This is followed by presenting large-scale agile context and some related Agile methods to the research findings. Then, literature related to the research findings and Agile architecture is presented.

2.2 Agile Software Development

Since the formulation of the Agile Manifesto, several iterative and incremental software engineering approaches have been emerged and hence fall under the umbrella of Agile software development [37]. These software engineering approaches advocate the philosophy captured in the Agile Manifesto. Some well-known examples of Agile software development approaches include Scrum, eXtreme Programming (XP), Crystal, Feature Driven Development (FDD), Dynamic Software Development Method (DSDM), and Adaptive Software Development (ASD). These approaches have some common shared

techniques such as iterative and incremental software development, communication, collaboration, coordination, acceptance of the uncertainty and welcoming change in requirements even at late stages of the project [111]. However, each Agile approach has its specific techniques, strengths and weaknesses [111].

Agile development methods are people-centred. They advocate an empowered and collaborative way of working that is very different from the traditional approaches to project management – such as the Waterfall and V-Model. Transforming organisations from traditional to Agile methods does not require only employing Agile practices but also readjusting the culture of organisations. Previous research [43] has identified 7 elements of Agile culture, which are employed in an Agile Culture Matrix. These elements are unleashed purpose and meaningful results, Agile leadership, well-being and fulfilment, collaborative communities and distributed authority, trust and transformation, adaptability to change, and innovative learning and personal mastery.

2.3 Agile Methods

This section presents some well-known Agile methods related to this research such as Lean, Scrum, and the Spotify Model.

2.3.1 Lean

Lean Software Development is a management philosophy that is a translation of lean manufacturing principles and practices to the software development domain [60, 117]. Lean Software Development has three pillars, (1) perfection of the product (maximising customer value), (2) perfection of the process (eliminating waste) and (3) perfection of the individuals (respecting people). Also, Lean has five principles, (1) identify value, (2) map the value stream, (3) create flow, (4) establish pull, and (5) seek perfection [75]. The applicability of Lean has been studied, and challenges are reported [75].

Because of the promising benefits that Lean affords, revised frameworks have been

proposed to employ Lean software development in large-scale contexts. For example, Kuusela and Koivuluoma [63] introduced a transformation framework, which includes three cycles: Strategic alignment cycle, organisational and business alignment cycle, and Lean implementation cycle. Another well-known example is the *Lean Startup* [14], which is introduced by Eric Ries [44]. Lean Startup emphasises the perfection of the product by developing what the customers want in a world filled with uncertainty, that is depicted in Minimum Viable Product (MVP). Lean Startup movement has adopted customer-centric software product development that is inspired by Lean principles.

2.3.2 Scrum

Scrum is an iterative and incremental Agile project management approach. This Agile framework consists of three components, including roles, ceremonies, and artefacts [102]. Also, Scrum comprises a range of common practices [85]. Scrum process includes 3 distinct roles: Scrum team, Scrum Master, and Product Owner (PO) [102]. Previous research [8] identified 9 roles for PO teams that are used to scale Agile methods to large projects. These roles are Groom, Prioritiser, Release Master, Technical Architect, Governor, Communicator, Traveler, Intermediary and Risk Assessor. Also, previous research [10] identified performed activities by POs as part of a product-owner team to manage scale, distance, and governance.

Some well-known ceremonies in Scrum include Daily Scrum Meeting, Daily Scrum of Scrums Meeting, Sprint Review Meeting, and Sprint Planning Meeting [102]. Also, Scrum provides 3 artefacts: Product Backlog, Sprint Backlog, and Burn-down Chart [102]. Paasivaara et al. [85] have identified 7 Scrum practices, which are typically used in global development, namely: Backlog, Sprint Planning, Sprint, Daily Scrum, Scrum-of-Scrums, Sprint Review (or Demo), and Retrospective.

Scrum-of-Scrums is a technique to scale scrum up to large-scale agile, where multiple teams work on the same project [66]. This technique is considered as an approach for coordinating multiple teams. In Scrum-of-Scrums, an ambassador from each team participates in Daily Scrum meeting. Each team decides which team member should be an ambassador to solve the inter-team dependency issues. Scrum-of-Scrums follow

an Agenda in which ambassadors report completions, next steps and impediments on behalf of the teams they represent. Resolution of impediments focuses on the challenges of coordination between the teams, such as interfaces between teams and negotiating responsibility boundaries. This technique facilitates cross-team synchronisation by facilitating inter-team collaboration and coordination, which in turn enhances teams productivity.

2.3.3 The Spotify Model

Spotify Organisation Structure: The Spotify model has been developed to facilitate the development of music streaming service in a very-large-scale organisation. Spotify organisation has hundreds of developers spread over more than 30 teams and across 3 cities [57, 58]. The Spotify organisation utilises an adaptive structure, which is based on a matrix of two dimensions (vertical and horizontal) that is weighted towards delivery and creativity [58, 57, 69]. The vertical dimension, which is the primary one, represents the physical location of cross-functional teams where they spend most of their time. The horizontal dimension is depicted in the knowledge sharing that would facilitate, support and remove the encountered impediments by the squads through the utilisation of Chapters and Guilds. Hence, while the vertical dimension represents the "What", the horizontal dimension represents the "How". While the entrepreneur (vertical) dimension tends to speed up the process, the professor (horizontal) dimension tends to slow down the process trying to build things properly. The overall structure of Spotify organisation consists mainly of Squads, Chapters and Guilds.

In the Spotify model, each squad is autonomous and has a long-term mission, which is based on the product strategy and a short-term mission that is quarterly renegotiated [57, 69]. Squad autonomy represents the ability of squads for making decisions on their own so that the company can scale without being held back by dependencies and coordination problems. The Squads are encouraged to implement Lean Startup principles such as the MVP where a feature is not finished until the impact is analysed [58, 69]. A squad does not have a formally appointed Squad Leader [57, 58]. The Squad Leader is responsible for communicating what problem needs to be solved and why. Hence, it is Squads' job to collaborate to find the best solution [56, 69]. A Squad has access to an Agile Coach; who

is in charge of helping squads to progress and improve their ways of working to achieve their goals [57, 69]. Also, a squad does not have a manager but rather a product owner who is responsible for prioritising the work [58, 69]. However, the Product Owner will not be involved in how to solve the problem. On the other hand, it is the responsibility of the Product Owners as whole to collaborate to maintain a high-level roadmap that shows where the organisation is heading [58, 69]. Moreover, maintaining a matching product backlog for each squad is also their responsibility [58, 69].

A Tribe is a collection of squads that are designed to be smaller than 100 people and aims to minimise dependencies that can slow or obstruct a squad. In order to promote collaboration between squads, they should be co-located. A tribe leader is responsible for providing the best possible habitat for all squads within the tribe [57, 58, 69]. A gathering is arranged regularly to show what squads have worked on, delivered and achieved so that others can learn from them [57, 58, 69].

Within the same tribe, there are small groups of people, called Chapters, that have similar skills and working within the same competency area. For example, a front-end engineer Chapter or quality assurance Chapter. People within a Chapter meet regularly to help in identifying and solving the problems, which can encounter other squads, quickly. Each Chapter has a leader who is responsible about communicating what problems need to be solved and why. A Chapter leader is considered as the line manager and works in a Squad as other members within the same Chapter. Chapters are considered the glue that sticks the company together without sacrificing too much autonomy [57, 58, 69]. While Chapters are always located within a Tribe, there are groups of people who are wide-reaching with a desire to share knowledge, tools, code, and practice across the whole organisation. These groups are called Guilds in Spotify [58, 116].

Previous research has identified 4 patterns for knowledge sharing by cultivating Spotify guilds [106]. These patterns are (1) Book clubs, which focus on learning rather than doing – utilising activities such as lunch and learn seminars, invited guests, and discussing better ways of working, (2) Open source societies, which concentrate on maintaining, improving, and setting the future strategy for specific components, (3) Support lines, which is responsible for onboarding new engineers, answering technical questions,

and facilitating solution discussions, and (4) Standardisation committees, which align practices across the company through establishing concrete artefacts such as coding standards or recommended tools.

Spotify Organisation Culture: The structure of Spotify organisation is designed to promote Agile development culture. In this culture, alignment of squads enables squad autonomy. Kniberg [57, 58] says: *"The stronger alignment we have, the more autonomy we can afford to grant... One consequence of having autonomous squads is having a very little standardisation"*. Hence, employing autonomous squads that are aligned increases motivation, quality and also fast releases. Each Squad has the autonomy to decide what to build, how to build it, and how to work together while building it by having the required support from Agile coaches. However, squads should be aligned to a product strategy. Thus; squads should be autonomous, but do not sub-optimise [57, 69].

Spotify engineering culture values cross-pollination more than standardisation [57, 58]. For example, if a specific tool gets used by many squads, that tool becomes a path of less resistance and others will tend to choose the same tool and thus became a default standard between all squads. Also, alignment over the product level is adopted because of the realisation of a very-large-scale organisation size where hundreds of developers are employed [58, 69]. However, In the Spotify culture, squads do share products instead of owning them.

Spotify aims to minimise handoff to scale out without having dependencies and coordination processes between squads [56, 69]. However, there will always be dependencies with multiple squads in place. Therefore, Spotify culture adopts some continuous discussions to find ways to eliminate problematic dependencies which often lead to reprioritisation, reorganisation, architectural changes or technical solutions [58, 69]. In Scrum-of-scrums, a synchronisation meeting is continuously conducted between ambassadors to report completions, next steps and impediments on behalf of the teams they represent [86]. However, in Spotify, a synchronisation meeting is allowed only on demand to coordinate between the involved squads where dependencies are existing [58, 69]. Hence, Spotify tries to minimise handoff and waiting to scale without having dependencies and coordination.

A fail-friendly environment is employed in the Spotify organisation to utilise a fast failure recovery strategy, which aims to speed up the learning process through capturing failures and eliminating them in future [56, 69]. In this fail-friendly environment, squads share encountered obstacles and failures through shared Fail Walls. In addition, a Postmortem Documentation process is performed at the end of each project by identifying what went successful or unsuccessful in mitigating future risks [56, 58, 69]. Thereby, the Spotify organisation improves the employed processes and the product itself by capturing learning lessons that can be used to avoid encountered problems in future.

The release strategy in the Spotify model is based on enabling decoupled releases by having a decoupled architecture [56, 69]. This strategy simplifies the release process and encourages squads to provide frequent small independent releases [69]. Thus, each client application or module has its release train that departs at a regular schedule [57, 69]. The squads are encouraged to release unfinished work as hidden to determine possible integration problems and minimise the need for code branching [57, 58]. In addition, Spotify employs a Limited Blast Radius technique to build an experimental friendly culture and to minimise possible negative ripple effects when encountering release failure [56, 69]. This Limited Blast Radius technique encourages squads to provide frequent small releases. Also, this technique does experiments on a limited number of end-users to get feedback and learn quickly, instead of wasting resources trying to predict and control all possible risks in advance.

2.4 Agile Architecture

In the last two decades, the role of software architecture, in Agile software development, has been perceived as a controversial issue [2, 16, 118]. While many advocates for giving to architecture the importance in Agile software development it has in other development approaches such as Big Design Up-Front, other opponents against that. These advocates have doubts about the scalability of any software development approach that does not pay sufficient attention to software architecture [2], especially for accomplishing required quality objectives when developing large-scale software projects.

Cockburn [21] argues about the unfeasibility of using Agile software development in large-scale and life-critical software projects. This unfeasibility is because the benefits of software architecture are missing in Agile methods and teams are completely dependant on tacit knowledge. However, Falessi et al. [33] found that Agile practitioners realise the software architecture as relevant based on some aspects, such as rationalisation of previous design decisions, documentation of ranked requirements that demand flexibility, and planning.

One of the principles behind the Agile Manifesto notes that *"continuous attention to technical excellence and good design enhances agility"* [37]. This principle indicates paying attention to the architecture makes Agile software development even working better. The essence of agility is responsiveness, learning, and sufficiency [16]. Neglecting certain architectural considerations even early in the development process can make architectural refactoring to be costly [31]. What teams build is influenced and constrained by how they build. Also, how teams build something is influenced and constrained by utilised design and architecture [16]. By focusing on a simple question, *"what architectural issues block a team's agility?"*, Agile practitioners can work together to achieve technical excellence, good design, and improve the agility of practitioners [16]. For instance, previous research [55] identified modular architecture and microservices as prerequisites for applying Agile practices. If the architecture is monolithic and a large part of it is built without modularisation, then applying Agile practices is considered as a difficult task [55].

Previous research found an iterative and incremental way of architecture evolution can reduce the Big Design Up-Front and keep the software project synchronised with the latest changing conditions [15, 27, 61]. Hence, several researchers realised the ability to have coexistence of software architectures and Agile development [2, 7, 15, 27, 31, 61]. Large-scale agile methods and frameworks, such as DAD [103] and SAFe [67], emphasise the necessity of architecture-agility combination and coexistence. This emphasis is depicted in the employment of Architecture Owners in DAD [103] and Enterprise Architects in DAD [103] and SAFe [67]. Each one of these roles has its identified responsibilities [67, 103]. This coexistence resulted in utilisation of architecting activities and approaches, as well as Agile architecting practices [118].

Yang et al. [118] found that the architecting process is comprised of 11 architecting activities, which cover the entire architectural lifecycle. These activities are Architectural Description, Architectural Evaluation, Architectural Understanding, Architectural Maintenance and Evolution, Architectural Analysis, Architectural Refactoring, Architectural Impact Analysis, Architectural Implementation, Architectural Synthesis, Architectural Reuse, and Architectural Recovery. These activities received varying degrees of attention in the architecture-agility combination. Most research effort has been put on Architectural Description since architecture has to be described to a certain extent based on the desires of Agile practitioners. Also, architectural understanding, analysis, and refactoring are considered as the most beneficial activities used with Agile development.

Also, Yang et al. [118] identified 41 Agile practices used in the architecture-agility combination. However, only a few of these practices have been widely used (such as "Backlog", "Sprint", "Iterative and Incremental Development", "Just Enough Architectural Work", and "Continuous Integration"). While some practices, such as "Iterative and Incremental Development", are commonly employed by various Agile methods, others are highly linked to specific Agile methods. Also, the employment of Agile practices is based on practitioners experience and knowledge because of the lack of guidance on how to use these practices should be combined in the architecture-agility combination.

Agile Architecture can be understood as patterns or tactics that enable a simultaneous focus on Agile development and architecture [11]. Bellomo et al. identified Agile architecture patterns that impact the employed resources (time and cost) to implement, test, and deploy requested changes. These patterns and tactics include separate interfaces, layer architecture, restrict dependencies and separate concerns. Also, Bellomo et al. identified some patterns that improve software product scalability. These patterns include encapsulation of algorithms, clustered architecture with load balancing and replicated copies, and data caching. Moreover, Bellomo et al. identified some patterns that focus on flexibility in deployment and controlling the cost and time for testing. These patterns include standardised and configurable architecture (parameterisation and static and dynamic binding), virtualisation (layering both the infrastructure and the application), and executable (interface-driven code structure).

An alignment of Agile architecture was proposed by providing an example of using architectural tactics and aligning architecture, production infrastructure, and Agile development teams [82]. Nord et al. explored architectural tactics that support scaled Agile development and improve the alignment of architecture and software development. They proposed an alignment approach that comprises a vertical and horizontal decomposition of software architecture as well as a matrix augmented-role team structures by using Scrum. They also proposed the employment of a catalogue of tactics mapped to Agile software development, which can be collected from empirical research and literature.

Architect roles and architecture practices were utilised to develop and evaluate a framework for Agile architecting embedded software products in large-scale organisations [73]. The identified Architect roles include Chief Architect, Governance Architect, and Team Architect. By analysing the relationships among the architects, Martini and Bosch found that most practices need such roles to coordinate and cooperate and hence mitigate the challenges. Also, they identified different sorts of teams: Feature Team, Runway Team, Architecture Team, and Governance Team. Feature Teams are steered by Product Managers and consists of cross-functional software development teams. Each Feature Team has a Team Architect who is responsible for the reference architecture and leading architecture activities. Runway Teams are dedicated teams for some identified sprints to focus on the architecture feature and hence to overcome architectural debt, which might lead to possible crisis. Architecture Team, which comprises the Architect roles above, work together to collaborate and coordinate as no single architect can have all the information needed to support numerous teams within a large-scale organisation. Governance Team consists of Governance Architects and Product Owners who are responsible for risk assessment of architecture changes and prioritising the backlogs of the teams to balance the short-term with the long-term objectives.

2.5 Agile Tailoring

Agile methods became increasingly attractive for many organisations of different contexts because of the realisation of successful implementation of projects using Agile

methods [17, 26]. Nevertheless, software development methods have been initially introduced for specific contexts with an assumption to be applicable for any kind of software project [17]. Subsequently, however, organisations encountered failed projects, and hence communities of Software Engineering started to research method tailoring to make the development methods more adherent to the development context (i.e., organisation or project level) [17, 18, 35]. Method tailoring is needed for either of organisation or project context adequacy despite the selected development method or process [18, 54]. A long time ago, software method tailoring had been discussed and continued to be a current need for the software industry [17, 35].

Software method tailoring is defined as *"the adaptation of the method to the aspects, culture, objectives, environment and reality of the organisation adopting it"* [17]. In the same way, Kalus and Kuhrmann [54] define software method tailoring as the adaptation of the development method to the project requirements. The implementation of software method tailoring requires strategies or approaches to execute the tailoring and might use a criteria to decide on how best to select tailoring options defined by the tailoring approaches [24].

2.5.1 Tailoring Approaches

The research work on Agile tailoring addresses different needs by using various approaches for tailoring. Previous research includes papers identifying most used practices [53, 62, 72], comparing identified Agile practices and quality levels [1], proposing new methods for Agile practices adoption based on different criteria [6, 19, 32, 46], considering maturity models to assess the adopted Agile methods and practices [3, 107], proposing hybrids of Agile and plan-driven approaches in large-scale setups [38, 45, 52], comparing different tailoring approaches [42], classifying the approaches used to perform method tailoring have been used in previous research [17, 24, 35].

Literature was surveyed on the use of Agile in global software development and then analysed to identify and listed the most referenced Agile practices [53]. Similarly, Kurapati et al. [62] surveyed the software development industry to understand which Agile

practices are being used. For a similar purpose, however, an empirical study identified common adopted Agile practices and listed practices that normally are used together to satisfy customers, by conducting a survey [72]. Manyam and Kurapati also tried to identify how the adaptation of the identified practices has been conducted. The commonly adopted practices that were identified include stand-up meetings, sprint/iteration, testing, communication, short releases and continuous integration.

Factor analysis was applied to a set of 58 Agile practices and extracted 15 factors or groups [1]. Abbas et al. claim that the identified factors and their associated practices can guide Agile process improvement. The correlation between the factors showed that using quality assurance, iterative and incremental practices increases the success rate.

The literature on situational method engineering was surveyed, and a synoptic evaluation was presented in the context of formalising a conceptual framework [46]. Henderson-Sellers and Ralyté proposed metamodels based on the surveyed results, which were then used to build high-level process models for method construction. This work also described method fragments/chunks by considering their identification and creation (from existing methods, from scratch or from past usage). Henderson-Sellers and Ralyté analysed method creation in terms of various processes for constructing a full methodology from method fragments/chunks.

The Strategic Analysis for Agile Practices framework [32] was proposed to facilitate the selection process for Agile practices adoption based on the business goals of the organisation. This framework is built upon a knowledge base of experiences collected from empirical studies, which extended the Situational Method Engineering concept and used concepts from the Balanced Score Card.

A metamodel for Agile methods constructions based on the concepts of Situational Method Engineering was designed [6]. Ayed et al. investigated how Agile methods can be constructed in-house to address specific software development needs by examining a case study, which focuses on the tailoring of XP and Scrum. This work focused on the customisation of any Agile method and investigated an approach based on the Situational Method Engineering paradigm, which includes measurement concepts for creating context-specific Agile methodologies. Then, Ayed et al. designed an Agile

metamodel to support the construction of new Agile methods. This metamodel relies on measurements to guide the construction of new Agile methods and throughout the development process. Also, this metamodel is based on assumptions originated from software development tailoring criteria, such as the organisation's business goals.

An approach for tailoring software development processes according to project needs was proposed [19]. This approach is built for a software product line and based on method engineering techniques. Thus, Casare et al. take into account the similarities and differences of adopted/tailored processes, as well as reusable process fragments.

Sidky Agile Measurement Index [3] was proposed as an Agile maturity model to facilitate the adoption of Agile practices. This model is a 5-step road map to guide teams based on essential values to agility: enhancing communication and collaboration (level 1), delivering software early and continuously (level 2), developing high quality, working software in an efficient and integrated manner (level 3), respond to change through multiple levels of feedback (level 4) and establishing an environment to sustain agility (level 5). The Sidky Agile measurement index is not based on a specific Agile method but instead uses Agile values and principles to define the path to agility.

An assessment framework was proposed to define which Agile practices and methods are appropriate for an organisation [107]. This approach examines Agile methods based on their adequacy, organisations capability to support the adopted principles and practices, and the effectiveness of the adopted Agile method. To facilitate the assessment process, Soundararajan et al. proposed the Objectives, Principles and Practices (OPP) Framework to guide the assessment.

The Hybridisation of Agile and plan-driven approaches should be scalable where such hybrids can provide better cost-benefit ratios [52]. Imani et al. conducted empirical research by conducting two case studies and surveys. They found that such hybrids are beneficial increases the likelihood of improving the cost-benefit ratio, compared to purely plan-driven methods. To benefit from such hybrids, Hayata and Han proposed an approach to project development and management by blending Scrum into traditional plan-driven project development and management [45]. Hayata and Han discussed the management activity involved in Scrum, investigated the team and meeting composing

of Scrum, analysed the challenges and benefits of applying Scrum in traditional project development and management, illustrated and discussed the blending structure, and compared the iterative process with Scrum and planned process without Scrum.

The integration of agile and plan-driven development elements was addressed by developing a hybrid adaptive methodology reference architecture model using a qualitative constructive empirical research approach [38]. Gill et al. conducted an iterative qualitative constructive empirical research to identify the agility, abstraction, business value, business policy, rules, legal, context and facility elements that have not been explicitly modelled or discussed in International Standards such as the ISO/IEC 24744 metamodel. Gill et al. claims that the elements of the hybrid adaptive methodology reference architecture can be used as a checklist or vision-guiding reference architecture when constructing various situation-specific agile and hybrid methodologies.

An empirical case study was conducted to compare Template-based and Automatic process tailoring approaches [42]. Template-based tailoring refers to a series of predefined processes for different project types and chooses the most appropriate one for each project. Automatic tailoring refers to a model-driven engineering approach for defining process models and adapting them to project contexts. González et al. applied both strategies to software process lines of a small Chilean company and found that automatic tailoring is always more productive, but this difference is mitigated when predefined processes are more frequently applied in the organisation.

Approaches to perform software method tailoring have been identified, in literature [17, 24, 35], and classified into Contingency Factors (aka Template-based Tailoring [42]) and Method Engineering (aka Automatic Tailoring [42]). The *Method Engineering* approach is based on meta-method processes and suggests the creation of new methods based on existing method fragments to be applied in specific contexts [17, 24]. This approach concentrates on all activities related to software development methods and creates in-house methods – organisation or project-specific methods [17]. These methods are more efficient to respond to challenges of real projects and their context instead of relying on existing methods [17]. This approach brings flexibility to the organisation but introduces challenges such as how to control method fragments or how to assemble

the new method for specific context and situations [17, 46]. The Method Engineering approach introduces other challenges because of the needs for (1) tools used for method construction, (2) quality evaluation models for newly introduced methods, (3) knowledge on how to handle specific situations – definition of fragments and specifying which fragment would work best in a specific situation [17, 46].

The *Contingency Factors* approach addresses the tailoring by selecting multiple methods to be available to the organisation and then performing method selection based on the features of the project context such as organisational structure and uncertainty level [35, 17]. This Contingency Factors approach assumes that there is no sufficient development method for all cases an organisation can encounter [24]. Thus, the right method should be selected from a portfolio of predefined methods based on the targeted development context and by following predefined criteria for method selection by the organisation [36]. The main challenge for adopting the contingency factors approach is that team members should have a good understanding of a range of software development methods and should be capable of executing them according to the contingencies needed for the context [36].

2.5.2 Tailoring Criteria

The process of software method tailoring can be influenced by tailoring criteria [17, 54, 114]. These tailoring criteria can be used to select which software development method or practices to be adopted by the organisations. Table 2.1 provides an overview of the tailoring criteria categories, their description, and some examples for each category.

Kalus and Kuhrmann [54] conducted a systematic literature review and identified 4 software development tailoring criteria categories. These categories are team, external environment, internal environment, and objectives. Campanelli and Parreiras [17] researched the tailoring criteria, by conducting a systematic literature review, to identify tailoring criteria used in Agile tailoring literature and hence expanded the categories defined by Kalus and Kuhrmann [54]. Campanelli and Parreiras proposed two new categories, called maturity levels and previous knowledge, and new attributes/items for

Table 2.1: The identified tailoring criteria categories, description, and examples

Tailoring criterion	**Description and examples**
Team	Represents aspects related to the development team and how the team is composed and interact. Examples: Size, distribution, turnover, domain, previous/good cooperation, tool knowledge, technology knowledge and process knowledge
Internal environment	Represents aspects related to the organisational operation and processes. Examples: Clear project proposal, prototyping, management availability and support, project budget, project type, project role, project duration, financial controlling, measurement, subcontractors, technical support, programming language, operating system, database, tool infrastructure, organisation size, communication aspects, and culture.
External environment	Reflects the aspects outside of the organisation Examples: Legal aspects, number of Stakeholders, stakeholder availability, stakeholder background, requirements stability, client process and availability, contract-type, user availability, user background.
Objectives	Represents the whole technological environment of the organisation as well as business goals to be achieved. Examples: Complexity, degree of innovation, domain, conceptual solution, technical solution, legacy system, legacy system documentation, safety and security, hardware development, user interface, system integration test, and business goals.
Maturity levels	Is concerned about tailoring software methods based on the maturity levels desired/needed by the organisations. Examples: Maturity levels to define which Agile practices should be adopted.
Previous knowledge	Considers the work done on previous projects and the team members previous experiences to drive the tailoring process

the remaining categories, which were identified by Kalus and Kuhrmann.

Tripp and Armstrong [114] conducted a survey to explore and investigate the relationships between the motives for adopting Agile method practices. The analysis of the drivers for Agile practices adoption and tailoring was conducted and identified internal environment and business objectives as the main factors affecting Agile tailoring [114]. Tripp and Armstrong identified 3 motives for Agile adoption: a desire for increased software quality, efficiency, or effectiveness, which are associated with different configurations of project management. Also, Tripp and Armstrong [115] identified 12 commonly used Agile development practices into a typology. This typology is based upon whether these practices are primarily of project management or software development approach focus. Besides, Tripp and Armstrong examined organisations' motivations for adopting Agile impact the adopted practices and hence causes Agile tailoring, which may lead (or not lead) to differences in project performance.

2.6 Large-scale Agile

At the early stages of the Agile software development introduction, Agile methods have been perceived to be the best fit for small collocated self-organising teams [30, 51]. Since many organisations with small collocated teams have realised successful implementation of projects utilising Agile methods, Agile methods became increasingly attractive for many other organisations with large-scale projects of different natures [17, 26].

The term large-scale Agile has been used to describe Agile software development in wide-ranging contexts. "Large-scale agile" is defined by Dingsøyr et al. [29] as *"Agile development efforts that involve a large number of actors, a large number of systems and interdependencies, which have more than two teams"*, which is in line with Rolland et al. definition [90]. Whereas, a very large-scale – *"Enterprise Agile"* [9, 34] – is defined as *"the Agile development efforts with more than ten teams"* [29]. In the same way, while Large Scale Scrum (LeSS) is applied to up to 10 teams (of around 7 people for each team), LeSS Huge is applied to a few thousand people working on one project [65].

Dingsøyr et al. [28] provided a taxonomy for scaling Agile software development. This

taxonomy is comprised of: (1) small-scale (1 team), (2) large-scale (2-9 teams), and (3) very large-scale (>9 teams). In this research, the term "large-scale project" refers to a project that employs Agile development and involves a large number of interdependencies and a large number of people including 2-9 collaborating teams of up to 100 members.

Many scaling Agile approaches exist in the literature, but only a few are found in the industry [77, 113]. These scaling Agile approaches include SAFe, LeSS, Scrum-of-Scrum, Nexus, or internally created methods [77, 84, 113]. Theobald et al. [113] reported the identified commonalities of existing scaling Agile approaches concerning their practices. They found that the existence of practices that are common to most Agile approaches and frameworks – like the scaled Scrum events, e.g., a scaled planning meeting or retrospective. Ozkan and Tarhan [84] conducted a review of scaling approaches to Agile software development methods (such as Scrum at Scale, Nexus, LeSS, SAFe, Spotify, DAD, and DSDM) and described how and to what extent these Agile methods provide scaling.

Large-scale projects are subject to challenges as several teams should work closely together to release a single software product while multiple teams need to communicate, collaborate and coordinate [23, 26, 79]. Moe et al. summarised some of the research discussions on challenges in large-scale Agile development [79]. Also, Conboy and Carroll identified some challenges associated with implementing SAFe, Scrum-at-Scale, LeSS, and other customised Agile frameworks [23]. The identified challenges in large-scale agile include maintaining teams autonomy [23] and aligning self-organising teams through coordination [26, 79], knowledge sharing [79], technical consistency [26] and other aspects [23, 26, 79].

2.6.1 Autonomous Teams

The concept of "Autonomous Teams" has different origins and definitions in the literature since it was studied and described from various perspectives in the past [110]. Stray et al. [110] found that the closest definition of autonomous teams as applied from outside software engineering into Agile development comes from the knowledge-management perspective. Stray et al. state that that the first introduction of autonomous teams into software engineering was made by way of the Agile manifesto, which cited self-

organising teams as the source of the best architectures, requirements, and designs. Other researchers define autonomous teams in terms of informal self-organising roles [50].

The concept of self-organising was recognised as one of the key principles of Agile software development since the introduction of Agile Manifesto [37]. Self-organisation is identified as one of the Agile success factors [20, 26]. Highsmith [47] describes Agile teams as self-organising teams, which are composed of *"individuals [that] manage their workload, shift work among themselves based on need and best fit, and participate in team decision making"*. Thus, autonomous teams should be composed of cross-functional teams that can cover the full development cycle of a project [55]. These Agile teams must have mutual trust, respect, common focus, and the ability to organise themselves repeatedly to overcome challenges [22].

Agile teams must have common objectives and should be accountable to commit to work and organise themselves to overcome encountered challenges [48, 55]. The autonomy of teams has a direct influence on teams' effectiveness since the authority of decision making is moved to the operational level, which increases the development speed and accuracy of problem-solving [76, 93]. Hoda [49] identified some practices of self-organising Agile Teams. Also, Hoda [50] identified six informal and spontaneous roles that Agile team members adopt, which are claimed to facilitate self-organisation and guide Agile coaches in working with Agile teams.

Autonomous teams are not uncontrolled – leaderless – teams [112, 22]. The leadership in self-organising teams is considered light and adaptive by providing feedback and subtle direction to the team [112, 4]. Leaders in Agile software development are responsible for building strategy, setting directions, obtaining resources, aligning people, and motivating teams [4]. These leaders have job titles in Agile projects – such as Scrum Master in Scrum, Coach in XP, and Squad leader in Spotify.

Previous research has identified some barriers to the autonomy of teams in large-scale agile. For instance, Moe et al. [78] found that (1) Scrum emphasises self-organisation but does not provide guidelines for how teams should be organised, (2) providing a conductive environment to the self-organisation increases external autonomy, and (3) high individual autonomy is considered as a barrier to self-organisation since members could prefer their

own goals over team goals. Also, interfacing, coordination, knowledge sharing, and alignment between teams are considered challenging [23, 26, 79]. Besides, organisations should employ sufficient autonomous release processes that grant independence and authority to teams [55]. This process helps individual teams to provide shapable new products or features in isolation from other teams.

The Spotify model is an example of an Agile engineering culture, which is driven by creating autonomous teams (i.e., squads) to facilitate software development for hundreds of developers [57, 58]. This model employs autonomous squads to promote sustainable creative development for a very-large-scale software development programme. The Spotify model is driven by creating autonomous yet tightly aligned squads [57, 71].

Mankins and Garton [71] identified 3 challenges to getting the balance right between individual autonomy and organisational goals when using the Spotify model. Firstly, balancing autonomy and accountability for produced results, and for the actions and behaviours that deliver those results. Organisations should put squads' strategy into practice with measurable objectives toward organisations' goals and specify clear expectations to reach the goals. Thus, squads are held accountable for achieving their missions since they have freedom in determining how to achieve their missions. Secondly, balancing freedom to innovate versus following proven routines to get both outcomes – consistency and innovation. In some software development areas, the speed of innovation is critical and demands autonomy, small teams, and organisational agility. Other areas, however, may benefit from standardised approaches of software development, which enforces routines. Finally, balancing alignment with control to ensure that coordination and connectivity happen among those squads without relying on controlling managers.

2.6.2 Inter-team Coordination

When teams grow in size, coordination becomes necessary to manage the dependencies between multiple teams. Coordination helps in accomplishing the transparent participation of all stakeholders and contribute to set rules and standards that would overcome possible difficulties that might hinder large-scale projects [28, 79]. Scaled Agile has been a hot research topic in the Agile community for some years now [28, 104]. Inter-team

coordination is one of the important topics that have been pointed out in large-scale agile in XP2016 [79]. Dingsøyr et al. [28] identified a standard taxonomy of Agile software development scale, which is connected to the coordination approaches. The small-scale agile, which consists of 1 team, use Agile practices such as daily meetings, common planning, review and retrospective meetings. The large-scale agile, which consists of 2-9 teams, achieves coordination in a new forum such as a Scrum-of-Scrums forum. The very-large-scale agile, which consists of more than 10 teams, employs several forms of coordination, such as multiple Scrum of Scrums.

In the last decade, large-scale software development has identified coordination challenges among teams. Hence, the team of the team's setup has been used because of the increasing dependencies, complexity and uncertainties [12, 59, 74, 101]. As projects grow in size and complexity, the number of inter-team dependencies also tend to increase [74]. Hence, more coordination effort is needed to deal with these dependencies so that each team's goal is achieved, and the overall goal of the project is obtained [59, 64, 86, 101]. Also, as specifications of the requirements might change over time, where Agile software development welcomes changing requirements even late in development, large-scale projects come with more uncertainties that also affect coordination. These uncertainties emerge because of the unpredictability of required tasks that team members shall perform [59]. For instance, what should be prioritised first, which team should tackle which tasks and how to do tasks, as different people might have different opinions. If inter-team coordination is weak, this can contribute to integration failure. Furthermore, with the team of the team's setup, the project increases fast in complexity [12]. Thus, large Agile projects do need coordination [26] to resolve the dependencies between different sub-projects and between the teams [70].

A typical example of handling inter-team coordination is depicted in the Scrum-of-Scrums approach [28, 101]. The team of team's setup addresses the needs for coordination as additional events that support cross-team coordination through; (1) inter-team Sprint Planning meetings, (2) inter-team Daily Scrums, (3) inter-team Product Refinements and (4) inter-team Sprint Reviews. These meetings are used as a way of keeping the teams synchronised and coordinated instead of ending up being a status report meeting for the management [64]. In these meetings, sending one representative of each team is

considered a common way to have sufficient meetings. However, Paasivaara et al. [86] emphasised the benefit of having smaller and focused inter-team meetings with only participants of similar goals and interests.

2.7 Summary

At the time this research study started, in 2015, there was no scientific research on the Spotify model. In 2017, however, Mankins and Garton [71] identified 3 challenges to getting the balance right between individual autonomy and organisational goals when using the Spotify model. Also, in 2019, Šmite et al. [106] has studied patterns for knowledge sharing by cultivating Spotify guilds. The critical analysis of previous research on the Spotify model has been limited to the grey literature [56, 57, 58, 69]. Also, there is no identified scientific research on the Spotify model across multiple projects, organisations, and cultures. Hence, this literature review identified a lack of scientific research into the Spotify model.

The Spotify model has become influential among Agile proponents. It formed the basis of Agile methods used in numerous other organisations of different contexts. This influence, in turn, led to the decision of exploring the Spotify model in other contexts than its originated (i.e., other software industries, projects, organisations, and cultures).

At the beginning of this study, a minor literature review was conducted to understand the basic facts and terminologies of the Spotify model and other research aspects to converse with the participants during interviews. However, a detailed literature review was conducted at later stages of this research on related aspects of the grounded findings to facilitate the discussion. Hence, this chapter presents an overview of the Spotify model and some related aspects to the findings such as well-known Agile methods, Agile architecture, Agile method tailoring approaches and criteria, large-scale agile, autonomous teams, and inter-team coordination.

Chapter 3

Research Design

3.1 Introduction

This chapter details the research approach of this study by following Creswell's framework [25]. Creswell's framework considers a Research Approach as an overarching framework that guides research studies from the philosophical assumptions behind an inquiry, which are followed by procedures of inquiry (called Research Designs), to detailed research methods for data collection, analysis, and interpretation. Research approaches are plans and procedures for research that involves several decisions on how to study a topic. The purpose of research approach selection is not only to guide the researcher based on the nature of the research problem but also to enable the audience to understand and evaluate the conducted research and its results. A research approach involves the intersection of (1) research philosophy, (2) research design (i.e., strategy of inquiry), and (3) specific research methods [25].

3.2 Components Involved in a Research Approach

Creswell [25] introduced a framework for research (as illustrated in Fig. 3.1), which is an interconnection between Philosophical Worldviews, Designs, and Research Methods. A research approach is defined by Creswell as *"the plan or proposal to conduct research, involves the intersection of philosophy, research designs, and specific methods"*.

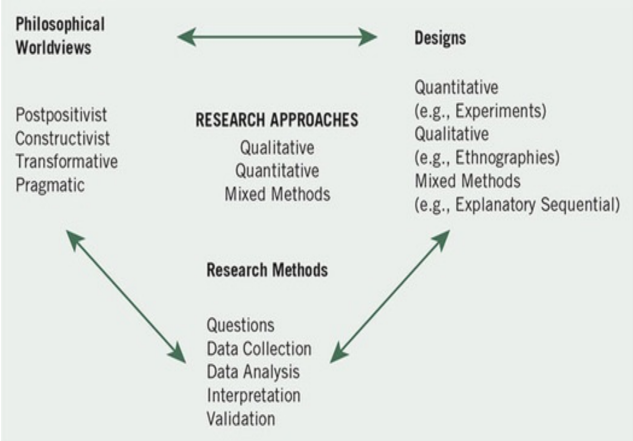


Figure 3.1: Creswell's framework for research [25]

Detailing an explicit followed research approach allows the researcher to situate the study in a scientific context. Hence, the audience will be able to understand the underlying philosophical assumptions and limitations, research objectives, and research results [25]. Creswell describes 3 framework components that involve a research approach:

- **Philosophical worldviews** are the assumptions about what constitutes knowledge claims.
- **Research Designs** are the strategies of inquiry, which are the general procedures of research.
- **Research Methods** are the detailed procedures of data collection, data analysis, and interpretation.

The following section describes the employed research approach by positioning these components in this study.

3.3 The Employed Research Approach

This research utilises a *qualitative design* [25]. In essence, this is the process of conducting a longitudinal embedded case study and then an intervention embedded case study. While the emphasis in this research is on the longitudinal embedded case study part, the intervention part provides additional details in response to some identified challenges. The outcome of this study is an analysis of the entire set of findings from both parts, which is an interpretation of the findings. This section describes the aims and objectives, philosophical worldview, research design, and research methods of the study.

3.3.1 Aims and Objectives

This research aims to (1) explore how Agile practitioners, from a FinTech organisation, resolve the conflicting trade-offs between squads autonomy and alignment, and (2) develop and evaluate new architectural governance practices, in a FinTech organisation tailoring the Spotify model. Based on these aims, I have identified an initial set of research objectives and questions at the beginning of this research study. However, the initial research questions have been changed because of the exploratory nature of this research study, which is concerned with theory generation. Thus, the initial research questions were changed to accommodate the emerged theory. The research objectives and questions are presented in their final state in Chapter 1 along to the detailed research motivation.

This study approaches the research problem through the identification of a research gap. This research gap was approached by identifying a lack of scientific research on the Spotify model. The research gap did not specify a precise problem statement or hypothesis. However, this study observed the existence of conflicting trade-offs between alignment and autonomy, which lacks scientific exploration.

The research problem area is approached with the assumption that software development is a social activity that is influenced by the employed processes, methods, tools, and hosted context (such as organisational culture and market). Hence, this research study is of exploratory nature since it tends to increase the understanding of the phenomenon of interest. The scope of this thesis is limited to theory generation, but theory testing in the sense of performing statistical tests or statistical generalisation to a large population is not the aim of this research study.

3.3.2 Philosophical Worldview

The philosophical worldview of research consists of claims about the nature of reality as what knowledge is (ontology), how researchers gained knowledge (epistemology), what values are related to knowledge (axiology), how knowledge is expressed (rhetoric), and what processes are employed to study knowledge (methodology) [25]. To determine how a phenomenon is approached, it is essential to have the ontological beliefs about the nature of reality and the epistemic relationship between the knower and the known. Addressing the ontologies and epistemologies inform the choice of methodology and channel how the research is shaped to collect and analyse data. Creswell discusses 4 philosophical worldviews or schools of knowledge: postpositivism, constructivism, transformative, and pragmatism.

This research study is situated in the field of empirical software engineering research. The researcher identified a lack of scientific research on the Spotify model. Hence, this research study aims to gain a thorough understanding of a phenomenon of interest within a particular context. The researcher studies the external world by basing the research on observation rather than relying on a purely analytical approach. However, a large part of the described phenomena in this study is not observed immediately in the workplace. The reasons behind observed outcomes exist in the mind of individuals. Hence, this thesis emphasises on understanding shared meanings of individuals that rule the interpretation of observed activities.

The lack of scientific research on the Spotify model prevented the researcher from studying antecedent conditions. Also, an absence of up-front clear research problem or

hypothesis, in this research study, dictated the employment of an inductive paradigm by harnessing a constant comparison of collected data at increasing levels of abstraction [39, 40]. Moreover, this research study is concerned with understanding reality and focuses on theory generation rather than verification. The worldview of this research study arises from studying employed actions, situations, and consequences rather than antecedent conditions. Therefore, the main contributions of this thesis can be situated as a *Pragmatic Worldview*.

3.3.3 Research Design

Research Designs refers to the strategies of inquiry, which are general procedures of research [25]. Research designs are considered as types of inquiry, within (1) quantitative, (2) qualitative, and (3) mixed-methods approaches, that provide direction for the operating procedures in a research study. The overall design for this research study is of *qualitative design*. This qualitative design consists of a longitudinal embedded case study and an intervention embedded case study.

A phenomenon needs exploration because it involves unstudied aspects or because it has no conducted scientific research on it. By the time this research study started, there was no scientific research on the Spotify model – to the best of my knowledge. Researchers can explore a phenomenon by conducting qualitative research using a case study or a grounded theory design [25].

Grounded theory is a design of inquiry in which the researcher derives an abstract theory of a process or action grounded in the views of participants. This process involves using multiple stages of data collection and analysis by harnessing a constant comparison of data at increasing levels of abstraction during which refinement and interrelationship of categories of information are derived [39, 40].

Grounded Theory helps in studying relatively new areas of research or when trying to gain a new perspective on a well-known area of research [39, 40]. The absence of up-front clear research problem or hypothesis in my research study demanded the employment of an inductive research paradigm. Grounded Theory recommends refraining from

formulating a research problem upfront [39]. Grounded Theory focuses on generating theories, rather than verifying existing theories. Hence, the employed qualitative design in this research study is of grounded theory.

3.3.4 Research Methods

Research methods are detailed procedures that involve the forms of data collection, analysis, and interpretation [25]. Data collection methods have different degrees of predetermined nature, such as the use of close-ended versus open-ended questioning. This variation, in turn, leads to a focus on numeric versus non-numeric data analysis methods. Creswell [25] and Oates [83] provide detailed descriptions of various research methods such as case studies, survey, and grounded theory.

In this study, I adopt a qualitative design using an approach informed by the grounded theory to facilitate the process of data collection and analysis. The following sections describe the employed Grounded Theory methods and procedures in this study. Since this research comprises two parts – longitudinal and intervention embedded case studies, the research setting, as well as the employed Grounded Theory methods and procedures, are described in the first part of this research study, which is Sect. 3.4.

3.3.5 Role of the Researcher

This research study carries out a Grounded Theory research, which employs an interpretive research approach. Thus, understanding the role of the researcher is considered essential to realise how the phenomenon under study is interpreted.

I completed a Bachelor of Science in Software Engineering from Al-Isra University, Jordan in 2005. Thereafter, I worked in the Jordanian software industry for 4 years, at Takarub for Telecom. Working as a developer in a customer-oriented organisation has exposed me to the inner workings of software development teams, management, and customers. During my job, I have experienced the move towards more Agile-like projects, which in turn increased my curiosity about Agile software development.

In 2009, I joined the Master program of Software Engineering at Chalmers University, Sweden. At the first year of my Master's degree, I was introduced to an Agile software development course. I got further interested in this area and therefore made my Master's thesis about Agile process tailoring [92]. Based on my strong academic record and good experience in the industry I was recruited at Volvo, in which I worked on embedded software systems for 3 years and a half.

In 2014, I decided to apply for a part-time PhD programme at the University of Salford because of my keen interest in conducting scientific research in Agile software development. I was admitted to the research programme and at the same time was working in the industry because of my passion for the practical experience in the industry. A few months later, in 2015, I move to work in the FinTech industry in Stockholm.

My long professional experience worked to my advantage in accessing organisations for participation in research. I was conscious at the same time, not to let this experience cloud the research. Therefore, I carefully approached observations and interviews with an open mind. In order to preserve consistency, I have personally conducted all data collection and analysis, with frequent feedback from my supervisors, peers, and industry practitioners.

3.4 Longitudinal Embedded Case Study

When I was admitted to the PhD programme in 2014, I was working at Volvo, which was not utilising the Spotify model at that time. Therefore, I decided to move to another organisation that employs the Spotify model to conduct my research study in while working as a senior software engineer. In 2015, I moved to a FinTech organisation (i.e., FinTechOrg), which have tailored the Spotify model. FinTechOrg is considered a market leader in the FinTech industry. Given the nature of my research aims and the decision to undertake a case study, the FinTechOrg in which I work in was identified as a successful candidate to conduct my research in since the targeted project was utilising the Spotify model for around 2 years before starting my research study. Thus, I have identified

and approached the managers or gatekeepers who have the expertise to recognise my research proposal once Human Ethics Committee approval was received (Appendix A). In result, the targeted organisation has agreed to conduct the case study, but after signing a confidentiality agreement. Thus, I was able to observe ceremonies of different sorts, had access to different kinds of artefacts and was able to search for participants by reaching out to practitioners with different roles and from different squads.

In order to explore and have a deep understanding of how the Spotify model is being used in other contexts than its originated, I have conducted a longitudinal embedded case study. Longitudinal case study research is particularly appropriate for studying complex and large-scale phenomena [91]. Also, longitudinal case study research is appropriate when the researcher is interested in the success of some activity/process or the success of a software project. Thus, I have observed different sorts of ceremonies over 21 months and observed Agile processes and interactions within 3 different squads (i.e., teams). Despite the benefits of longitudinal case study research, it is considered expensive and time consuming for PhD students.

Being embedded in FinTechOrg was beneficial to understand the complex case of how Agile practitioners balance squad alignment and autonomy. During this longitudinal case study, I moved between 3 different squads. Researchers make a distinction between holistic case studies, in which the case is studied as a whole and embedded case studies where multiple units of data collection and analysis are studied within a case [91]. Holistic case studies take a broad perspective on the case and consequently may not be able to examine sufficient detail in the case and may, therefore, miss important issues. Also, holistic case studies are inappropriate for complex cases and for cases that need investigation over an extended period of time. However, embedded case study design is more appropriate under such situations since embedded design anticipates the need to collect, analyse, and report on complex detail in the case.

Conducting embedded case studies allow researchers to collect more instances of each of the embedded units of analysis, in contrast to the small number of holistic cases [91]. I have had access to different kinds of artefacts, such as documentation, product backlog, tools, and used technologies since I was embedded in FinTechOrg. This triangulation

in data collection and analysis has provided me with different angles towards the studied phenomenon and thus provided a broader picture. This, in turn, helped me as a researcher to overcome the complexity in the studied case and provided me with a deep understanding of the phenomena.

This longitudinal embedded case study lasted over a period of 21 month, from February 2017 to Mars 2019. The research findings related to this longitudinal embedded case study part are described in Chapters 4-7.

3.4.1 Research Setting

This research study was undertaken in a multinational organisation of very large-scale, which is a market leader in the FinTech industry. The headquarters located in Stockholm, Sweden, in which around 200 staff members are working in its software development organisation. The organisation employs approximately 650 staff members, including support and management, and they are distributed into 60 markets. This FinTech organisation provides in-store and online payment solutions as well as fraud protection tools. The organisation processes around 60 billion € per year.

The focus of this study is only on one project that manages autonomous online payment services. These autonomous software systems are a collection of payment solutions, which operate under the control of one administrative online software project. This project presents a standard and clearly defined management policy to the FinTech service. A Business-to-Business (B2B) model is employed to provide other companies (i.e., customers) with a FinTech service to perform a business functionality that is outside customers' main business domain. Consequently, customers do not need to invest in integrating their projects into hundreds of payment service providers (such as PayPal, Trustly, Entercash, ApplePay, WorldPay, or Klarna) around the word but rather into a single project that manages such complexity, which is the case study project. The case study project has a product-line that facilitates integrating the project into external payment APIs, which are provided by payment service providers.

From the customers' point of view, this project is classified as an offshore outsourced project, based on the proposed terminology and taxonomy by Šmite et al. [105] for global software development. The customers are distributed around the world in this B2B environment. These customers rely on FinTechOrg to integrate the project into many payment service providers' APIs as well as providing state-of-art FinTech services and features. Hence, FinTechOrg is located outside of customers' national boundaries (i.e., offshore) and allows customers to outsource the financial project as they do not have the expertise and they do perform a business activity that is outside FinTech industry.

The case study project is considered of large-scale, based on the proposed taxonomy by Dingsøyr's et al. [28, 29] for Agile development scale. This is because the development programme in FinTechOrg has 2-10 teams with less than 100 people. The software development programme is co-located in the headquarter and has 6 squads (i.e., teams) consisting of around 37 developers. Besides the developers, there are 1 test lead, 1 architect, 3 Key Account Managers, 5 Product Owners (2 of them are empowered with Key Account Manager role), and 2 Agile Coaches.

The early stages of the case study project, FinTechOrg was using Lean. This software development framework was powerful since this project contains a product-line. Some Lean principles adopted by the organisation include Kanban board, continuous process (Lean Thinking), documenting the development process (value stream mapping), and automation whenever possible. In late 2014, however, the organisation realised the needs for improving teams' autonomy and hence improve innovation. Therefore, the organisation decided to adopt the Spotify model, which was introduced by Kniberg and Ivarsson [56, 57, 58].

3.4.2 Data Collection

Data were collected through face-to-face semi-structured open-ended interviews, direct observations of Agile practices, documentation, and artefacts of different kinds – such as product backlog, tools, and technology. Collecting data from multiple sources brought a richness of data through triangulation [91]. Appendix A shows the ethical approval letter

and Appendix B presents the interview guide for data collection.

An iterative process of data collection and analysis was adopted to conduct a constant comparison of data. Performing constant comparison of collected data facilitated the guidance of future interviews and the analysis of interviews while observations fed the emerging results [40, 91]. As the data was analysed and new concepts and categories emerged, the subsequent interview questions had minor updates to focus on the emerging codes.

The observed ceremonies include backlog grooming, planning sessions, retrospectives, daily stand-ups, POs synchronisation meetings, and meetings of the whole project teams. The employment of direct observation provides researchers with a deep understanding of the studied phenomenon and prevents suspected deviation between "semi-structured interviews" view of matters and the "real" case [89].

The interviews targeted participants from different areas of software development (such as development, support, management, and customer representatives) to determine their perception of Agile tailoring when using the Spotify model. Thus, practitioners in several different organisational roles were approached, such as Agile coach, senior developer, Product Owner (PO), Key Account Manager (KAM), and Architect. The interviewees were provided with the opportunity to raise other issues through an open-ended guide [80]. The questions were revised after the second interview to adapt the questions to focus on emerging concerns. Also, the questions were revised to choose participants that can provide information on the emerging concerns. Each interview was recorded (approximately 50 minutes) and then transcribed verbatim for detailed analysis on a continuous basis.

In the longitudinal embedded case study part, the data were collected through multiple sources, including interviews, observations, and artefacts. This researcher conducted 14 face-to-face semi-structured interviews. Table 3.1 depicts the distribution of the interviewee's roles. Also, the researcher observed around 225 ceremonies over 21 months. Moreover, the researcher had access to different kinds of artefacts, such as documentation, product backlog, tools, and used technologies.

Table 3.1: The distribution of interviewee's roles - longitudinal embedded case study

Roles	Count
Agile Coach – Architect	1
Product Owner	3
Key Account Manager	2
Product Owner empowered with Key Account Manager role	2
Senior developer	5
Support manager	1

3.4.3 Data Analysis

This study analysed the collected data using an approach informed by the Grounded Theory method (Glasserian approach) [39, 40]. The Grounded Theory method has a set of principles and practices that constitute methods, which consist of systematic yet flexible guidelines, for collecting and analysing qualitative data to generate theories. According to Andrews et al. [5], the methods used in the Grounded Theory offer a "rigorous, orderly guide for theory development". The main features of the classic Grounded Theory method, which are followed in this thesis, are iterative and constant comparative analysis, open, selective and theoretical coding, memoing, and theoretical saturation [39, 40].

Why Grounded Theory (Glaserian approach)?

The Grounded Theory, which was initially established by Glaser and Strauss [13, 41], has now different approaches because of the schism that occurred between the two founding authors on how the methodology should be applied [13]. The main difference between their approaches is associated with the emergence and forcing of categories [13]. While the Glaserian approach promotes the identification of participants' differing perspectives at an abstract level and conceptualises them to find a latent pattern, the Straussian approach encourages the development of categories under directions (conditions, context, interaction strategies, literature or the researcher's experience). However, both

approaches emphasise the importance of constant comparison of concepts that will lead to conceptualisation (Glaserian approach) or descriptions (Straussian approach).

The divergence away from the original classic Grounded Theory [41] has led to two distinct schools of Grounded Theory; Classic Grounded Theory (Glaserian approach) [39, 40] and Straussian Grounded Theory (Straussian approach) [109]. Discussing the conflicting claims made by Glaser and Strauss about the different approaches or versions of the grounded theory is outside the scope of this study. However, the critical engagement with the philosophies behind the grounded theory is considered essential to understanding the implications that arise from a researcher's relationship to data collection and analysis.

The strength of grounded theory emerges from its ability to provide in-depth insight into how meaning is navigated and integrated within a specific context [13, 41]. The grounded theory facilitates the construction and sharing of the ontological belief about a phenomenon. Also, employing the grounded theory facilitates the determination of the influencing epistemic theories of knowledge. Moreover, Glasserian (i.e., classic) approach [40] emphasises the importance of iterative and constant comparison of concepts that will lead to conceptualisation to find a latent pattern while having an absent up-front clear research problem or hypothesis. Hence, the grounded theory (Glasserian approach) was identified as an appropriate research methodology for this study.

Main Procedures of the Grounded Theory

The following procedures outline the specific methods that were implemented throughout data analysis. While implementing these methods, the researcher adhered to the main features of classic Grounded Theory (Glaserian approach) [39, 40]. The employed research process is illustrated in Fig. 3.2.

Minor Literature Review: This minor literature review was conducted to understand the basic facts and terminologies of the Spotify model, which is the research area of interest, and other related aspects to converse with the participants during interviews. Hence, a minor literature review was conducted at the beginning of the case study, as

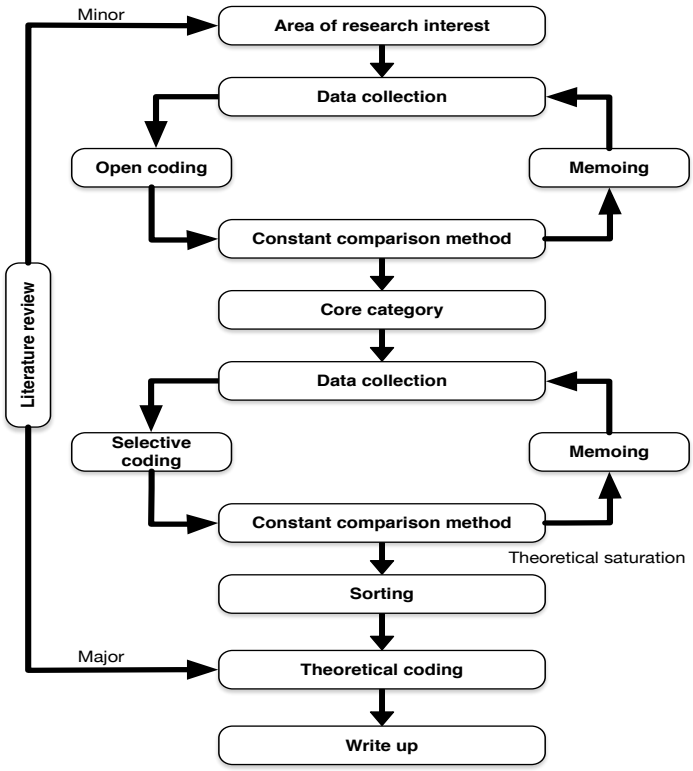


Figure 3.2: Research methodology process. Adapted from [49]

recommended by Glaser [40]. Sect. 2.3.3 and Sect. 2.4 shows the outcome of the conducted minor literature review about the Spotify model and Agile architecture. A deeper understanding of the Spotify model came from the observation, and the participants since the grey literature were the only source.

Open coding: As soon as some data have been collected, the data analysis can begin by employing a Coding mechanism. To analyse the collected data in detail, an open coding mechanism was used to break down the collected data analytically in detail [39, 40]. This mechanism begins by collating key points from each interview transcript. Then, a code, which represents a phrase that summaries the key point in 2 or 3 words, is assigned to each key point. A few questions were asked while conducting this open coding, as suggested by Glaser to make effective coding without being overwhelmed by the data [40].

Constant comparison: A constant comparison method, which rigours the grounded theory, is employed after conducting each interview [39, 40]. This method involves a constant comparison of emerging codes from each interview against other codes from the same interview as well as those from other interviews and observations. Using this constant comparison method again facilitates the process of grouping emerged codes into a higher level of abstraction, called concepts in Grounded Theory. Repeating this method on the emerged concepts facilitates the creation of third level of abstraction, called categories in Grounded Theory.

Fig. 3.3 illustrates a model that depicts the emerging levels of abstraction when using the Grounded Theory. This model is used throughout this research study to present the grounded findings, as illustrated in Fig. 4.2, Fig. 5.2, and Fig. 6.2. In this model, the **Categories** are marked with bold text and located within the rectangle. Each **Category** is associated with multiple *Concepts*, which are marked with italic text. Each *Concept* is supported by multiple Codes, which are marked with plain text.

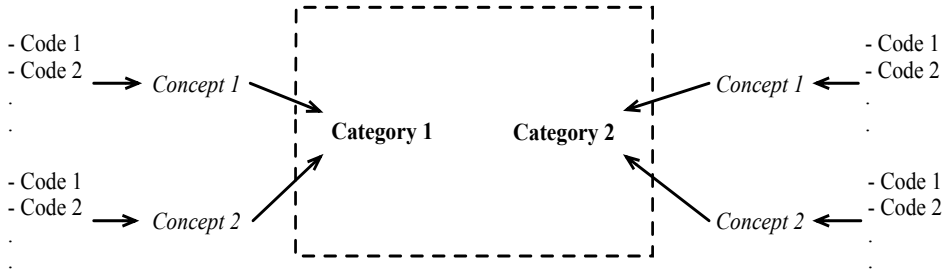


Figure 3.3: A model illustrates the levels of abstraction using the Grounded Theory

Core category and selective coding: The core category must be central and related to several other categories and their properties where it frequently occurs in the data and must account for most variations in data [39]. The constant comparison is repeated every time a new category is found or modified, which leads the research to revisit previously coded transcripts to check if they have a new property. When the core category is identified (such as the "Adaptive structure"), the researcher ceased the open coding process and moved into selective coding [39, 40]. Selective coding involves selecting codes related to the core category by limiting the coding to those variables

(concepts or categories) that relate to the core category in significant ways as to produce a theory [39, 40].

Memoing: An ongoing process of writing memos is employed throughout the grounded theory, called Memoing [39, 40]. The written down memos represent ideas about emerging codes and their relationships. Memoing is considered as a powerful way to pour out the emerging variables (codes, concepts, or categories). Also, memoing facilitates the emergence of relationships (similarities or differences) between different concepts and later between different categories. Moreover, continuous data collection and analysis reflects on memos' ideas and causes some modification.

Sorting: Sorting the theoretical memos can be initiated by the researchers when data collection is almost finished, and coding is almost saturated. Sorting collected memos (ideas) produces a theoretical outline, which puts the scattered data back together [39, 40]. This produced outline of the theory represents how the different categories relate to the core-category.

Major Literature Review: The literature on the Spotify model as well as other related aspects to the findings was reviewed extensively after the findings were sufficiently grounded and developed. The primary purpose of having a major literature review after the analysis is to (1) protect the emerging findings from preconceived notions and (2) to discuss the research findings with relation to the literature [39, 40].

Theoretical Coding: Theoretical coding involves conceptualising the relationships between the emerged categories and how these categories can be integrated into a theory [39, 40]. For example, 3 main aspects related to the Spotify model were identified in the first part of this research study. These aspects are (1) establishing and building autonomous squads, (2) aligning autonomous squads, and (3) performing B2B product development by tailoring the Spotify model. Revealing these aspects facilitated the identification of a novel approach to Agile tailoring, called *"Heterogeneous Tailoring Approach"*, by looking at Agile tailoring through the lens of the Spotify model.

Writing up: Following the Grounded Theory method has led to the generation of a substantive grounded theory of Heterogeneous Tailoring approach. The final step in the

Grounded Theory method is writing up the theory, which follows the theoretical outline generated as a result of sorting and theoretical coding. The write up is presented in Chapters 4-8.

3.5 Intervention Embedded Case Study (Action Research)

In the Software Engineering discipline, case studies often take an improvement approach, similar to action research [91]. This part of my study has a flavour of action research since it aims to intervene in order to resolve the challenge aligning and governing Agile architecture across autonomous squads. Whether to label the enquiry an *intervention case study* or *action research* is, to some extent, a matter of taste [91]. Throughout this research study, I refer to the improvement approach or the action research as the *intervention embedded case study* unless explicitly specified otherwise.

The most popular types of actions in action research in software engineering are Direct Interventions and Indirect Intervention [108]. Direct Interventions introduce changes to the company's operations directly. Then, the practitioners change their ways of working and collect the data to evaluate the change. In Indirect Interventions, researchers influence practitioners to make the change themselves. The researchers present new analysis results and solutions to practitioners who decide whether to adopt them or not. Indirect Interventions is more specific to software engineering, where the researchers are embedded in software development organisations. Hence, the direct interventions make a change, whereas the indirect interventions cause the change to happen (or a reflection to happen, sometimes the change is rejected). My intervention embedded case study is of Indirect Intervention type since I have influenced the practitioners to make the change by presenting my analysis results and highlighted on the encountered challenges.

The action research process can be defined as a number of research cycles, which are illustrated in Fig. 3.4, consisting of phases that involve diagnosing, planning, taking, evaluating the action, and learning [108]. The action research cycle starts with *diagnosing*, which refers to the identification of problems and the possible underlying causes. *Action*

planning specifies the anticipated acts that can solve the problems. *Action taking* refers to the implementation of the planned actions. *Evaluating* is the assessment of the intervention. Finally, *learning* is a reflection on employed activities and produced outcome.

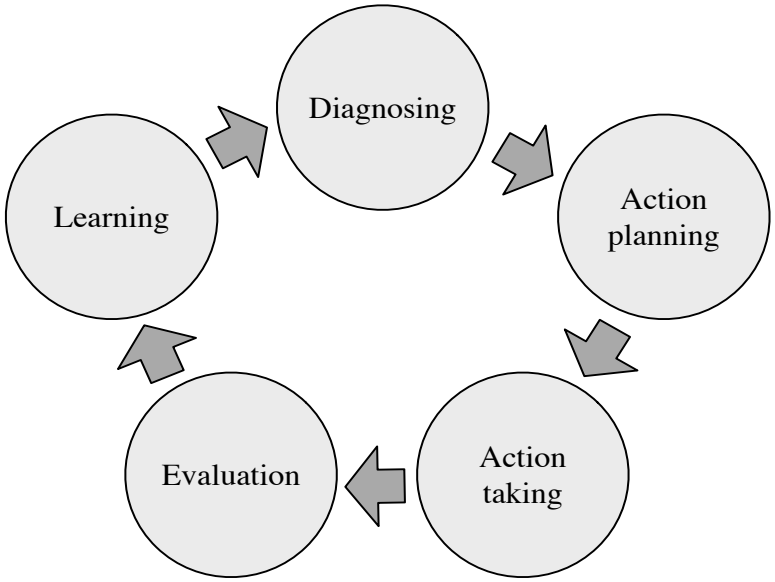


Figure 3.4: The action research cycles. Adapted from [108]

3.5.1 Diagnosing the Problem and the Underlying Causes

The findings of the longitudinal embedded case study part, which are presented in Chapter 4-7, identified two main challenges to the Heterogeneous Tailoring approach. These identified challenges the underlying causes behind them are described in Sect. 7.4. Hence, I decided to intervene and resolve one of the identified challenges, which is aligning and governing Agile architecture across autonomous squads.

3.5.2 Action Planning

In order to overcome the challenge of aligning and governing Agile architecture across autonomous squads, I decided to develop an approach to architectural governance by tailoring the FinTech Spotify model and evaluate it in FinTechOrg. Thus, I developed an approach to architectural governance, which incorporates a structural change and an architecture change management process.

3.5.3 Action Taking

The intervention embedded case study was conducted in the same research setting in which I conducted the longitudinal embedded case study. The research setting is described in Sect. 3.4.1. Initially, I presented the developed approach to architectural governance to the Agile coaches of FinTechOrg. During this presentation, I clarified the required changes to the organisational structure and the necessity of architecture change management process, which I have developed.

The Agile coaches agreed to conduct an intervention in one squad and two chapters. Thus, I presented my developed approach to the squad and then the squad started utilising the introduced approach to architectural governanace. This intervention embedded case study lasted 3 months, from November 2019 to March 2020. During this period, some improvements were introduced by FinTechOrg into my approach to architectural governance.

3.5.4 Evaluating

During this intervention, I conducted a direct observation of Agile practices during which 32 ceremonies were observed. To understand the reasons behind the occurred results, I collected the personal experiences of the practitioners after the intervention. Hence, 8 semi-structured open-ended interviews were conducted. The distribution of interviewee's

roles for newly conducted interviews in this research part is presented in Table 3.2.

The collected data was analysed using an approach informed by the Grounded Theory, as described in Sect. 3.4.3. The development team of FinTechOrg was positive towards my approach for architectural governance, as described in Chapter 8.

Table 3.2: The distribution of interviewee's roles - Intervention embedded case study

Roles	Count
Agile Coach	1
Enterprise Architect	1
Product Owner	1
Senior developer	3
Chapter Leader (Architect Owner)	2

3.5.5 Learning

The research findings related to the intervention embedded case study part are described in Chapter 8. I describe the benefits and challenges of the evaluated approach to architectural governance. Also, I reflect on the Heterogeneous Tailoring approach by adapting it to accommodate the evaluated approach to architectural governance.

3.6 Summary

This chapter presented the adopted research approach in this research study by following Creswells' framework for research [25]. Thus, the employed philosophical worldview, research design, and research methods in this study were presented. Consequently, this study utilised *qualitative design*. The philosophical worldview of this study was situated as a Pragmatic. The employed qualitative design in this study utilised mainly the Grounded Theory. The research design in this study comprises two parts: a longitudinal embedded case study and an intervention embedded case study (action research).

In both parts of this research study, data were collected through semi-structured open-ended interviews and supplemented by direct observation and access to different kinds of artefacts. In the longitudinal embedded case study, 225 ceremonies were observed over 21 months, and 14 semi-structured open-ended interviews were conducted. In the intervention embedded case study, 32 ceremonies were observed over 3 months and 8 semi-structured open-ended interviews were conducted. The collected data was analysed using an approach informed by the Grounded Theory method (Glaserian approach).

Chapters 4-7 describe the outcome of the conducted longitudinal embedded case study. Chapter 8 describes the outcome of the conducted embedded case study intervention.

Chapter 4

Spotify Tailoring for Establishing and Building Autonomous Squads

4.1 Introduction

This chapter addresses the research question (RQ1.1): *How do agile practitioners, from a FinTech organisation, establish and build squad autonomy?* To answer this research question, a longitudinal embedded case study was conducted, as described in Sect. 3.4.

The findings described in this chapter identify factors for establishing and building autonomous squads. The notion of establishing and building autonomous squads is illustrated in Fig. 4.1, which represent the bottom side of the whole model illustrated in Fig. 7.2. Fig. 4.1 shows the hierarchy of squads and how they tailor different agile methods to improve their autonomy. This chapter is an expansion of a peer-reviewed research paper [98].

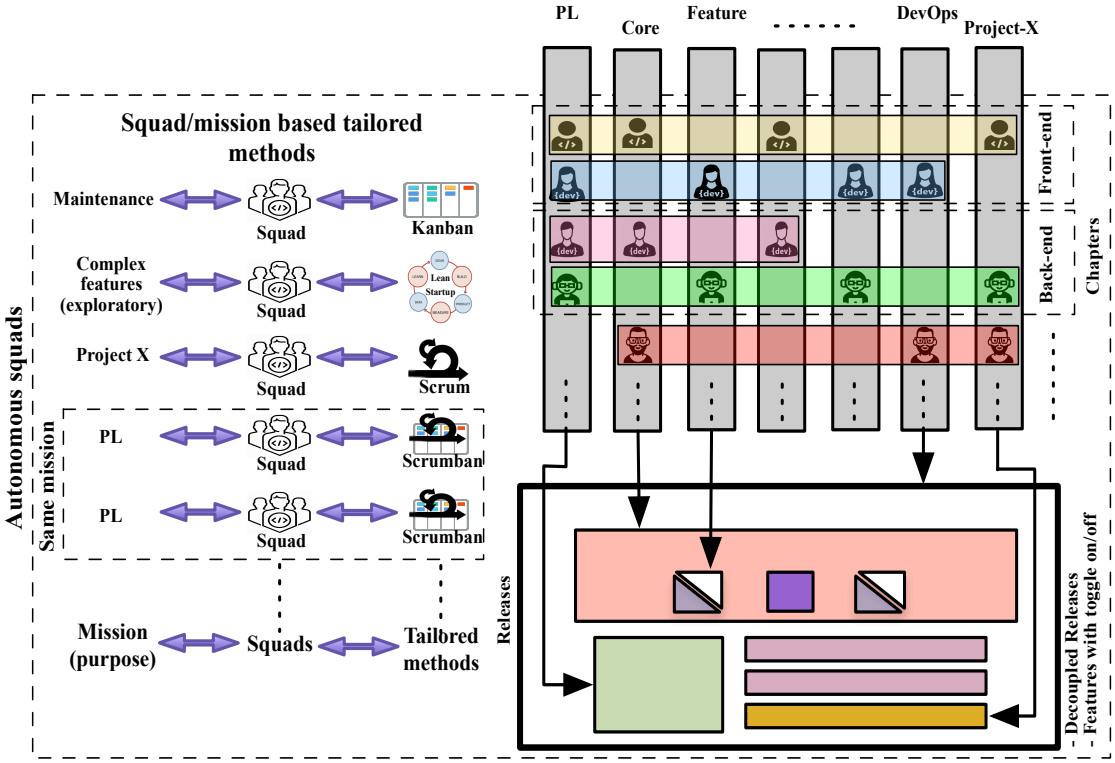


Figure 4.1: Autonomous squads

4.2 Influential Factors on Establishing and Building Autonomous Squads

The analysis of the collected data has grounded a synergy between the following categories and establishing and building autonomous squads. The emerged categories are (1) Employing adaptive structure, (2) identifying and allocating squad or mission-based strategy, and (3) employing squad or mission-based agile method tailoring. These emerged categories represent factors than influence establishing and building autonomous squads. Fig. 4.2 depicts these emerged categories along to their related concepts and codes. This figure is built by following the model illustrated in Fig. 3.3, which presents the emerging levels of abstraction when using the Grounded Theory.

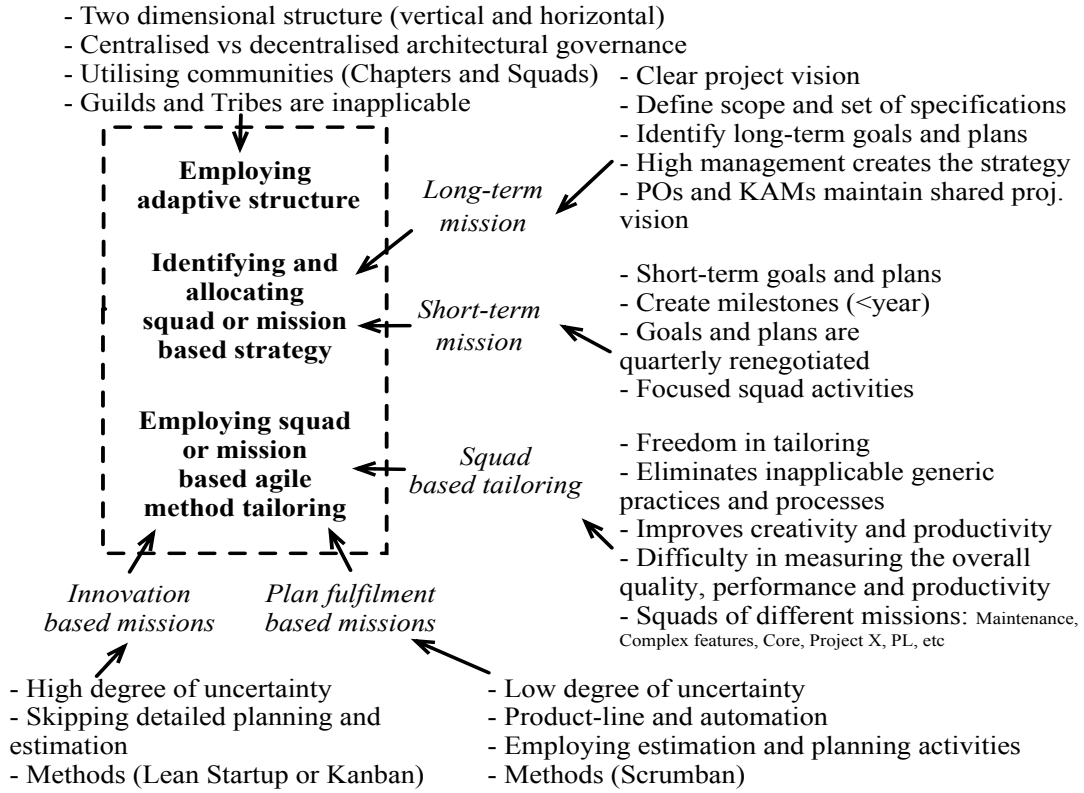


Figure 4.2: Emergence of the categories that facilitate establishing and building autonomous squads, which are dapted from [98]. The **categories** are in bold, the *concepts* are in italic, and the codes are in plain text

The following subsections describe the emerged categories and their related concepts and codes, which are depicted in Fig. 4.2. These categories represent influential factors on establishing and building autonomous squads.

4.2.1 Employing Adaptive Structure

FinTechOrg utilises Spotify's adaptive structure to establish autonomous squads, as illustrated in Fig. 4.3. The analysis of collected data shows the employment of cross-functional squads since each squad has members of different skill sets. Also, squad members are located within Chapters based on their competency areas. *"Each squad*

has members of different skills where these members are distributed over Chapters based on their competency area."–P10, Product Owner. However, it was observed that some members are located in more than a Chapter. A practitioner says: *"I work as full-stack developer and located in two Chapters"*–P8, Senior Developer.

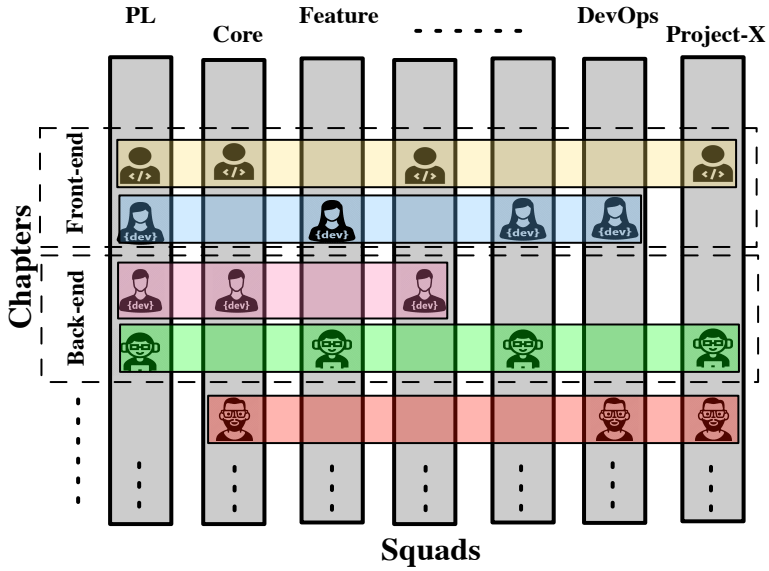


Figure 4.3: Two dimensional structure

Chapters communities facilitate decision-making across squads. *"The members of my Chapter meet regularly, and whenever is needed to help in solving encountered issues or make decisions on how to build things in a standardised way"*–P4, Senior Developer. Nevertheless, an architect role was employed because of the complexity of this case study project. Practitioners say: *"Deciding about which solution to adopt is time-consuming for the developers despite the employment of Chapters communities "*–P1, Agile Coach and Architect. Also, *"we discuss newly requested features with the architect or those developers who have experience in specific parts of the project to find the best solution"*–P10, Product Owner. Moreover, *"I get in touch with our architect whenever is need to discuss some user stories and to decide on which architectural change should be employed"*–P9, Senior Developer.

It was observed that Chapter leaders do not usually handle architectural based decisions within their Chapters since architectural decisions are centralised and granted to the

Architect. Also, the findings show a lack of process that could be employed to facilitate aligning and governing enterprise architectural decisions. A practitioner says: *"We do not have a process that aligns architectural decisions among the squads"*–P3, Senior Developer. This lack of process impacts negatively the autonomy of the squads. Also, it impacts negatively the quality being produced while working towards achieving common technical or business roadmap.

Tribes and Guilds communities were found to be inapplicable for FinTechOrg because of the size of the development programme. While the software development programme in Spotify itself is of very-large-scale (>300), the development programme in this case study project is of large-scale (<100). A practitioner says: *"While Tribes are inapplicable, building Guilds causes a waste of time and resources since the size of the development programme is smaller than Spotify's"*–P7, Product Owner and Key Account Manager. Thus, FinTechOrg replaces Guilds by informal and on-demand knowledge sharing. *"Meetings are arranged through email or Slack to tackle interesting subjects in different areas... Anyone interested in the subject can join"*–P12, Product Owner. This process, in turn, replaces Guilds communities with a sufficient process while preserving the autonomy of squads.

4.2.2 Identifying and Allocating Squad or Mission Based Strategy

Squads work autonomously to accomplish the strategy and roadmap of the organisation. The findings show that the strategy of FinTechOrg has emerged from its mission in the FinTech industry, which demands having a clear vision. A practitioner says: *"Fulfilling our mission in this industry demands an establishment of a clear vision, which provides a good description for what to be achieved"*–P1, Agile Coach and Architect. Establishing a clear vision, in turn, facilitates the creation of long-term goals for the organisational strategy. These long-term goals can be broken down into short-term goals. Thus, each squad in FinTechOrg has two types of missions that should be fulfilled: long-term mission and short-term mission. A practitioner says: *"The overall roadmap is represented in a collection of actions, which are employed to achieve the long-term and short-term goals of our squads"*–P7, Product Owner and Key Account Manager. Hence, the employed

activities seek to fulfil the organisational mission and guide strategic decisions while preserving squads autonomy.

Long-term missions are based on the organisational strategy, project vision, and long-term goals. It was observed that the organisation establishes a clear project vision to provide squads with clear directions and avoid confusion during the development. Practitioners say: *"Project vision is frequently communicated to make sure that all squads are working toward achieving their long-term goals"*–P12, Product Owner. Also, *"to establish clear project vision, the scope is defined, and a set of specifications are provided to the squads"*–P5, Product Owner. Hence, *"each squad work toward achieving identified long-term goals, which represent our overall organisational strategy and vision"*–P1, Agile Coach and Architect.

The findings show that maintaining a shared project vision and specifications demands conducting regular meetings between Product Owners and Key Account Managers. In these meeting, the stakeholders ensure (1) maintaining a shared project vision, (2) the ownership of the project since the organisation provides a software service, and (3) that all squads work toward achieving the long-term goals and strategy. Practitioners say: *"We make sure that product development does not deviate from our organisational vision... The customers' try to push software service development in their way. Therefore, we ensure the ownership of the project itself"*–P6, Product Owner and Key Account Manager. Also, *"in our regular meetings, we discuss the vision to ensure preserving the highlighted trends and directions by the top management"*–P13, Key Account Manager. Besides, *"resolving potentially conflicting priorities between the squads ensures squads' alignment over the organisational mission"*–P10, Product Owner.

Short-term missions represent the conversion results of squads' long-term missions and goals into short-term goals and plans. It was found that the squads' tactical objectives (i.e., short-term goals and plans) represent the building blocks for squads' long-term goals and hence represent the organisational roadmap and strategy. A practitioner says: *"Utilising the determined milestones strengthens squads' autonomy and provides us with short-term motivation and guidance to develop valuable software features that would satisfy our customers' needs"*–P5, Product Owner. These short-term missions might take

less than a year to be completed, and they are quarterly renegotiated. *"We (i.e., POs) meet every couple of months with the top management to review our milestones... We highlight what is achieved to discuss and prioritise what remains"*–P10, Product Owner.

The findings show that squads' autonomy demands an awareness of the expected outcome from each squad. Product Owners embrace long-term goals to create short-term goals and plan to compete in the marketplace. A practitioner says: *"Project vision, which provides long-term directions, is broken down by POs into milestones that should be achieved by the squads to compete in the market"*–P6, Product Owner. Achieving the organisational strategy is tightly linked with achieving squads' strategy (short-term goals and plans). *"Accomplishing the milestones supports our walking toward achieving the roadmap"*–P12, Product Owner. Identifying what was accomplished facilitates the measurement of success through the determination of which building blocks (i.e., short-term goals and plans) have been achieved.

4.2.3 Employing Squad or Mission Based Agile Method Tailoring

The analysis of collected data reveals that FinTechOrg employs squad or mission-based agile tailoring, as illustrated in Fig. 4.5. Each squad, in FinTechOrg, has the freedom to independently tailor agile methods based on its mission while having required support from agile coaches. Practitioners say that *"our squad has the freedom to tailor agile practices to fit our squad's needs... This way, we have a development process that suits our squad, and this consequently speeds up the development process"*–P8, Senior Developer. Also, *"adapting squads' processes based on their missions increases the ability to incorporate squad or mission-based processes... This adaptation mitigates the needs of handling general practices and processes that could be challenging for some squads and their missions"*–P1, Agile Coach and Architect. Hence, *"squad-based agile tailoring improves our productivity and creativity since we adapt our practices to be compatible with our mission and needs"*– P12, Product Owner. Also, *"adapting our squad's practices to conform with our mission eliminates the employment of inapplicable general practices, speeds up our development process, and hence facilitates accomplishing our own mission"*–P3, Senior Developer. Thus, the employment of squad or mission-based

agile tailoring improves the creativity of some squads and the productivity of other autonomous squads, preserves squad autonomy, and mitigates possible challenges of handling generic practices and processes.

The findings show that missions are mainly classified into either innovation or plan fulfilment nature based on the missions of squads. However, missions vary in their degrees of innovation and plan fulfilment, as illustrated in Fig. 4.4. The squads in this case study project have a variety of missions such as features, mini-projects, Product-Line (PL) for provider integration, maintenance, DevOps, etc.

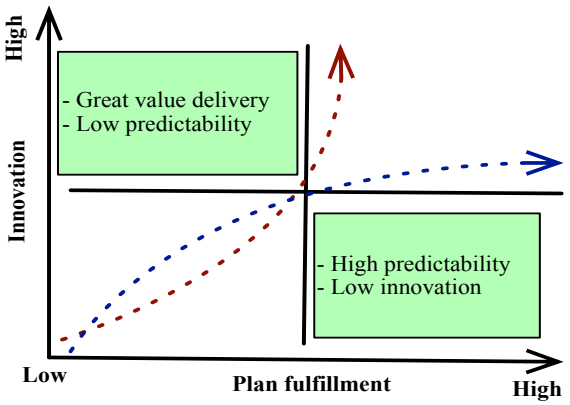


Figure 4.4: The nature of squads missions

It was observed that some missions value innovation more than plan fulfilment. Whereas, other squads value plan fulfilment more than innovation. Thus, each squad performs agile tailoring based on the nature of its mission, as illustrated in Fig. 4.5. The findings show that squads with innovation-based missions do utilise a tailored process based on (1) Kanban, (2) Lean startup or (3) skipping detailed planning and estimation activities. Also, the findings show that squads with plan fulfilment based missions – PL – do utilise a tailored process based on process automation and Scrumban.

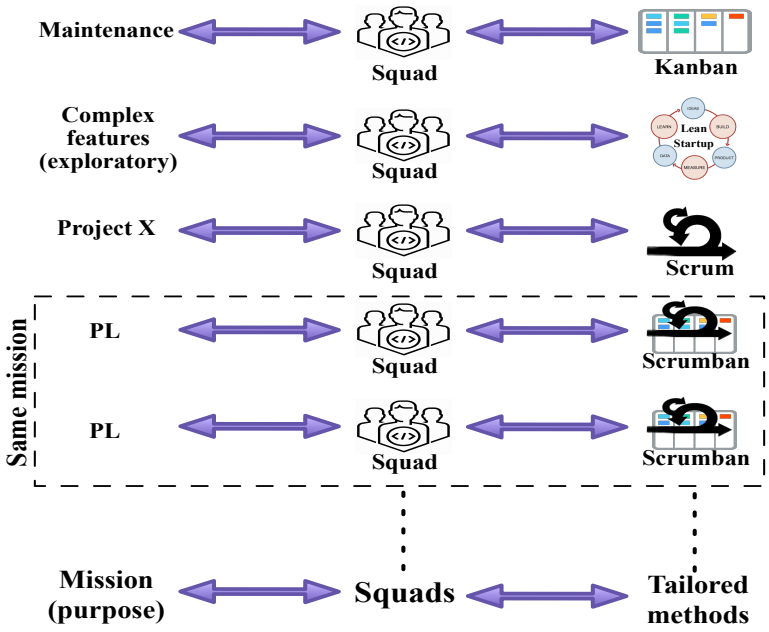


Figure 4.5: Squad or mission based tailoring

Innovation-based missions: The analysis results show that squads tackling maintenance based user stories, which are of adaptive or perfective nature, do employ Kanban. Practitioners say: *"maintaining the behaviour of already existing features can be challenging because of the complexity of the software service... We use Kanban to manage our work and balance customers' demands with available capacity of resources"*–P12, Product Owner. Also, *"providing our customers with online Kanban boards gives them sufficient insight about the work progress"*–P10, Product Owner. Hence, low coordination is employed, and better management decisions are taken continuously at the right time.

Also, it was observed that squads working on newly requested features, which are characterised with a high degree of uncertainty and complexity, do employ a tailored process based on Lean Startup and Kanban. A practitioner says: *"A hybrid process based on Lean Startup and Kanban are employed to facilitate the development of new generic and complex features"*–P3, Senior Developer. Hence, such squads can continuously develop and improve such features to reach the desired impact and hence satisfy customers.

Moreover, it was found that squads responsible for developing newly formed mini-

projects do employ a tailored process based on the Scrum method. Such mini-projects require innovation and take 9-18 months of development to introduce newly developed modules. Since such projects are characterised with a high level of uncertainty, estimation process and planning activities are either tailored or eliminated to mitigate any possible waste. Practitioners say that *"we do not use story points to estimate the effort required to deliver stories. Yet, we report the spent time"*–P9, Senior Developer. Also, *"the burn-down chart is used to check remaining work in the sprint backlog... We focus on providing valuable software product rather than planning"*–P10, Product Owner.

In addition, Software development estimation is considered as a waste for innovation based-missions. It was found that predictability of delivery is sacrificed for the sake of achieving innovation. Practitioners say that *"we sacrifice predictability of delivery to provide valuable software features to our customers"*–P6, Product Owner and Key Account Manager. However,*"customers request sometimes estimation before starting the development... We provide a rough estimation and keep them involved in the process to revise the plan accordingly"*–P12, Product Owner.

Plan fulfilment based missions: The organisation utilises a PL architecture since the case study project manages autonomous FinTech services. A practitioner says: *"Since our project manages autonomous financial services, a PL architecture is employed to facilitate the process of integrating the project into external APIs"*–P2, Senior Developer. Thus, it was found that the organisation classifies these external FinTech services into different categories using predefined specifications and requirements. This classification facilitates the automation of software development processes. It was observed the employment of predefined checklists to facilitate the development in the PL. *"Predefined checklists are employed in our PL to facilitate the process of integrating the project into external APIs... these checklists cover many aspects such as code review, documentation, estimation, security aspects, knowledge sharing, etc."*–P4, Senior Developer

Software development estimation is considered beneficial for those squads working on the PL. The analysis revealed that the predictability of delivery is beneficial for PL based tasks because such tasks are characterised with a low degree of uncertainty. A practitioner says: *"A detailed documentation is provided by third parties (financial*

software service providers) to facilitate the process of integrating the project into the targeted APIs"–P4, Senior Developer. It was observed that squads working on the PL do utilise a tailored process from Lean and Scrumban. A practitioner says: *"In the planning meeting, we use bucket size, on-demand planning techniques, and average lead/cycle time"*–P5, Product Owner. Thus, it was found that Product Owners are more confident in promising customers with predictable delivery deadlines because of being able to obtain the average lead/cycle time for different kind of user stories within the PL. *"Calculating the average cycle time for different kind of API integrations within the PL enables predicting the delivery and facilitates the planning... Consequently, we can promise our customers with delivery deadlines"*–P5, Product Owner.

4.3 Summary

This chapter has addressed the research question (RQ1.1): *How do agile practitioners, from a FinTech organisation, establish and build squad autonomy?* A longitudinal embedded case study was conducted to answer this research question. The data were collected through observing 225 ceremonies over 21 months, conducting 14 semi-structured open-ended interviews, and accessing different sorts of artefacts. The collected data were analysed using an approach informed by the Grounded Theory method.

The analysis, in this chapter, identified influential factors on establishing and building autonomous squads. These factors are (1) employing adaptive structure, (2) identifying and allocating squad or mission-based strategy, (3) employing squad or mission-based agile method tailoring. Those organisations wishing to build autonomous squads using the Spotify model can benefits from these factors. To put it succinctly, organisations should tailor the Spotify's adaptive structure and create appropriate communities of practice around this structure based on the development programme size. Those created communities of practice should have identified strategy. Each autonomous squad has a specific mission to accomplish, which comply with an identified strategy. However, multiple squads can share the same mission. Based on the assigned strategy, squads can tailor agile methods to fit their needs best and hence to accomplish their mission.

Chapter 5

Spotify Tailoring for Aligning Autonomous Squads

5.1 Introduction

This chapter addresses the research question (RQ1.2): *How do agile practitioners, from a FinTech organisation, achieve and sustain the alignment of autonomous squads?* To answer this research question, a longitudinal embedded case study was conducted, as described in Sect. 3.4.

The findings described in this chapter identify factors that influence the alignment of autonomous squads. Fig. 5.1 illustrates the alignment notion among the squads, which is depicted in the middle side of the whole model illustrated in Fig. 7.2. The alignment represents an umbrella that defines a set of practices, which facilitates the alignment of autonomous squads while progressing toward achieving their missions.

Investigating Spotify Tailoring, from a squad alignment perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. This investigation facilitated the identification of the FinTech's Spotify model, from squads

alignment perspective. Also, the investigation of Spotify tailoring has identified influential factors on aligning autonomous squads. Each identified factor is supported by a set of practices and attributes that strengthens the alignment of autonomous squads. The identified influential factors and their related practices can aid agile practitioners in aligning autonomous squads.

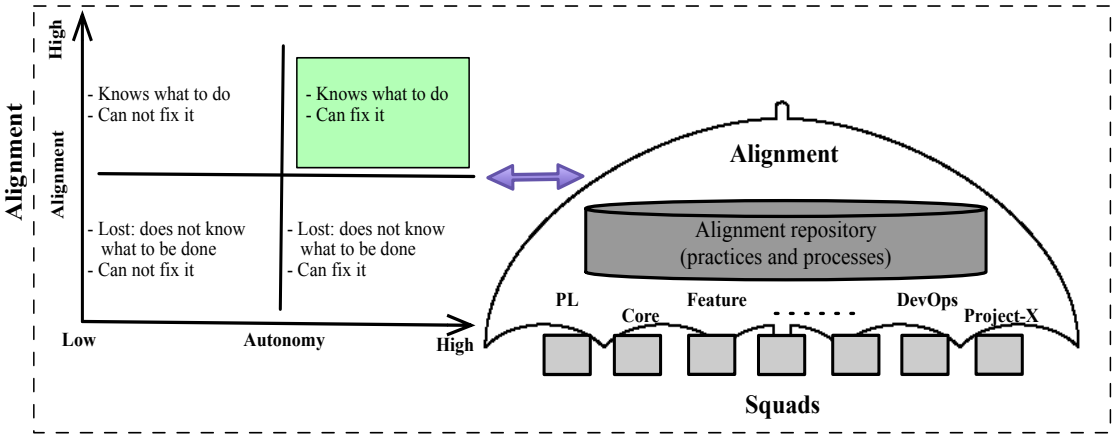


Figure 5.1: Aligning autonomous squads

This chapter is an expansion of two peer-reviewed papers [93, 94]. The first paper [93] identified influential factors on aligning Spotify squads. The second paper [94], investigates Spotify Tailoring, from squads alignment perspective, by identifying new, modified and previously introduced practices and attributes to the Spotify model. Besides, the second paper [94] identifies 3 new influential factors on aligning autonomous squads in addition to those published in [93].

5.2 Spotify Tailoring for Aligning Autonomous Squads – Practices and Attributes

This research study identifies practices and attributes that facilitate the alignment of autonomous squads. These practices and attributes promote effectiveness in cross-functional autonomous squads. The identified practices and attributes are tagged by α, β,

and γin Fig. 5.2. "α" represents practices and attributes that are previously introduced by the Spotify model. "β" represents Spotify's practices and attributes that were modified by FinTechOrg. "γ" represents newly introduced practices and attributes by FinTechOrg.

In result, *31* tailored practices and attributes, from a squad alignment perspective, were identified. These practices and attributes, which represent the emerged Codes in the Grounded Theory, were classified into three main types, as follows:

1. *Already exist: 14* practices and attributes, which are enumerated in Table 5.1, were followed by FinTechOrg as previously introduced by the Spotify model. These practices and attributes are tagged by α in Fig. 5.2.

Table 5.1: Already existing practices and attributes in the Spotify model, from a squad alignment perspective

Already existing practices and attributes
Two dimensional structure and employing Chapters and Squads communities
Product-level alignment
Define roadmap, scope and specifications
Definition of Awesome
Chapter based decision-making
Routine squad-of-squads meeting
Routine demos
Postmortem Documentation
Peer code review on the squad level
Decoupled architecture
Decoupled releases
Frequent small releases
Limited Blast Radius
Release train

2. *Modified: 5* adapted Spotify practices and attributes by FinTechOrg,, which are enumerated in Table 5.2. These practices and attributes are tagged by β in Fig. 5.2.

Table 5.2: Adapted Spotify practices and attributes by FinTechOrg, from a squad alignment perspective

Adapted practices and attributes
Guilds and Tribes are neglected
Overcome lack of knowledge and expertise – such as employing code review and more structure around the ownership
Chapter based knowledge sharing
Limited Fail-friendly environment (fail-wall)
Shall not release unfinished code

3. *New: 12* new practices and attributes, which are enumerated in Table 5.3, were introduced by FinTechOrg. These practices and attributes are tagged by γ in Fig. 5.2.

Table 5.3: New practices and attributes introduced by FinTechOrg, from a squad alignment perspective

New practices and attributes
Cover the value chain
Cover the whole lifecycle
Prevent hands-off and improve confidence
Utilising product-line and standardisation (checklists)
Lean thinking
Management commitment
Stakeholders should possess good experience and domain knowledge
Informal and on-demand call for sharing or discussing subjects
POs routine meetings
Non-routine meetings to control the outcome of intercorrelated tasks
POs' informal and on-demand meetings to resolve conflicted priorities
On-demand releases

5.3 Influential Factors on Aligning Spotify Squads

The analysis of the collected data grounded a synergy between the categories (i.e., factors), which are depicted in Fig. 5.2, and strengthening the alignment among the

squads. Fig. 5.2 depicts the emerged categories along to their related concepts and codes. This figure is built by following the model illustrated in Fig. 3.3, which presents the emerging levels of abstraction when using the Grounded Theory. This section ignores the category of *"adaptive structure with more focus on communities"* since it has been described already in Sect.4.2.1 and the literature [56, 57, 58].

The identified practices and attributes, which are presented in Sect. 5.2, are classified into the 7 grounded influential factors on aligning autonomous squads. Table 5.4 presents this classification. Also, Table 5.4 indicates the coverage of the adopted practices and attributes by the Spotify model and FinTechOrg. Moreover, the table clarifies the extent to which the Spotify model has been scaled in FinTechOrg (i.e., Spotify Tailoring).

Table 5.4: Spotify Tailoring for aligning autonomous squads

Factors	**Concepts**	**Spotify**	**Case Study**
Adaptive structure	Two dimensional structure	Yes	Yes
	Utilising Chapters and Squads	Yes	Yes
	Utilising Guilds and Tribes	Yes	No
Collective code ownership	Alignment over the product-level	Yes	Yes
	Adopting a reconciliation process	Unknown	Yes
	Strengthening PL automation	Unknown	Yes
Decision-making	Shared understanding of business objectives	≈Yes	Yes
	Knowledge-based decision-making	≈Yes	Yes
Inter-team coordination	Formal inter-team coordination	No	Yes
	Informal inter-team coordination	Unknown	Yes
Knowledge sharing	Systematic knowledge sharing	≈Yes	Yes
	On-demand knowledge sharing	≈Yes	Yes
Mission based planning	Innovation missions embrace Lean Startup	≈Yes	Yes
	PL missions embrace standardisation	No	Yes
Release strategy	Speed up the release delivery	Yes	Yes
	Accountability for release delivery	Unknown	Yes

≈**Yes**: partially covered, **Yes**: covered, **No**: not covered, **Unknown**: no evidence

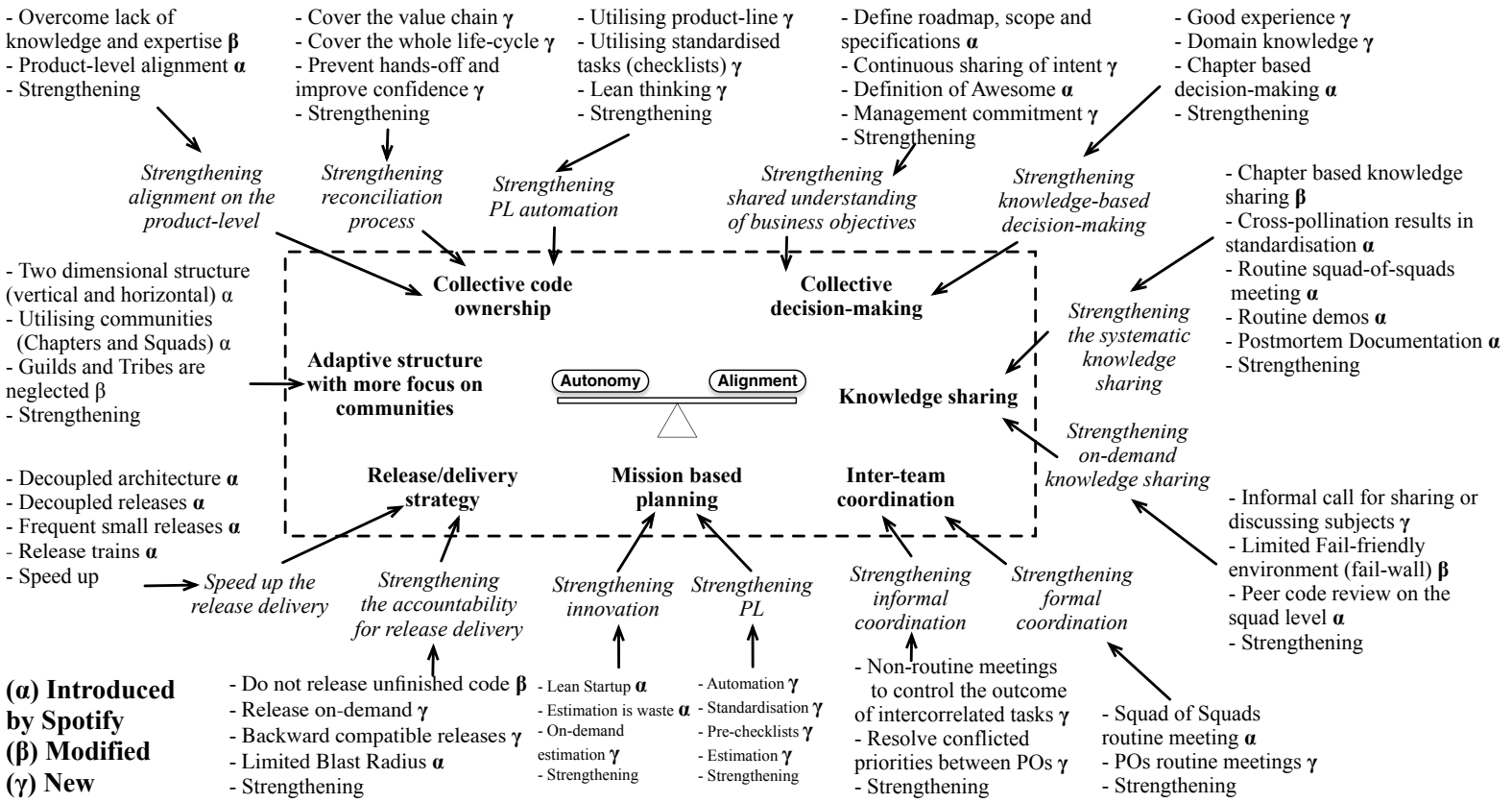


Figure 5.2: Emergence of the influential factors on aligning Spotify squads, which are dapted from [93, 94]. The **categories** are in bold, the *concepts* are in italic, and the codes are in plain text

The following subsections describe the emerged categories and their related concepts and codes, which are depicted in Fig. 5.2. These categories represent influential factors on aligning autonomous squads.

5.3.1 Collective Code Ownership

Strengthening alignment on the product-level: The squads are empowered to do the required software development on different associated software systems because of the realisation of collective code ownership. However, the organisation has realised that sharing products instead of owning them causes a waste of resources. This waste might happen because of a lack of knowledge on the product-level or because of insufficient ownership. Practitioners say: *"Handling maintenance or improvement tasks associated to the product-line by other squads are considered as time-consuming and will likely require relearning to be able to take the right action"*–P1, Agile Coach and Architect. Also, *"Encountering conflicting priorities between the squads may demand handling some work items that are outside a squad's expertise. Handling such work items requires some learning to gain the needed knowledge to proceed in development"*–P12, Product Owner.

Having many working items of discrete nature, in which each user story requires one developer to work on, impacts knowledge sharing and causes a waste of time and resources. This impact on knowledge sharing, in turn, results in a need for employing a relearning process. Practitioners say: *"Maintenance tasks are of discrete nature since only one developer is needed to work on each user story"*–P6, Product Owner and Key Account Manager. Thus,*"peer code review on the squad level is important to share knowledge and to ensure the successful development of user stories"*–P8, Senior Developer. Thereby, aligning the squads on the product-level supports collective code ownership, and facilitates the process of knowledge sharing and mastering.

Strengthening reconciliation process: Sharing a software product among autonomous squads requires a reconciliation process between the key associated parties, and before employing the desired change at the software level. A practitioner says:*"The squads*

might need, sometimes, to discuss the proposed solution with their peers, management team or architecture"–P3, Senior Developer. The main reason of this is either a lack of expertise at the product-level or the realisation of the product as a service where the organisation needs to be taken into account the complete value chain and the whole lifecycle. Otherwise, *"the task might get blocked, or a waste of resources might be a result of implementing an inefficient solution or shortage in the commitment of management or third parties"*–P3, Senior Developer.

Despite the utilisation of Chapters and Squads communities, it was observed that developers, sometimes, tend to be unconfident when working on a shard product that is not within their expertise. For instance, when encountering an incident where a hot-fix is requested, the reviewers (who are either from other squads or the same squad) were hesitant to handle the situation. This hesitation is mostly because of either being uninformed about the low-level details or the high complexity of such tasks, which requires knowledge transformation and relearning. A practitioner says: *"A Hot-Fix is requested in case of facing incident after a new release. If was not possible because of the absence of the developer, the release is rolled-back. Mostly, the developer who owns this task is requested to solve the issue with some support from the squad"*–P6, Product Owner and Key Account Manager.

Strengthening PL automation: Adopting a PL requires task standardisation to facilitate the process of software development. This standardisation, in turn, aids other squads working on related aspects. Since the organisation is providing a software service, which manages autonomous offshore software systems, it adopts a PL to integrate the project into external APIs. A practitioner says: *"Employing a PL architecture facilitates and speeds up the process of integrating our software system into external APIs"*–P2, Senior Developer. The lifecycle of the PL is illustrated in Fig. 5.3.

A task standardisation, which is a key principle in Lean Thinking (eliminating the waste), was observed through the utilisation of predefined checklists. These predefined checklists are utilised to facilitate requirement elicitation, technical details such as a security-related checklist, planning, estimation, documentation, code review, and knowledge sharing. Hence, these checklists help FinTechOrg in speeding up the process and eliminating

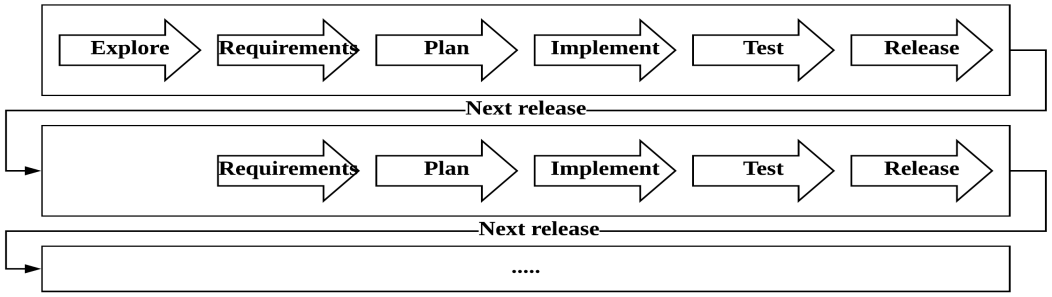


Figure 5.3: Product-line lifecycle [93]

possible waste. However, the employed checklists need to be enhanced further, and other potential areas need to be covered. Practitioners say: *"Technical related predefined checklists (such as security and code review checklists) are missing some details... Enhancing these checklists and creating new ones to cover other types of tasks would improve PL development process"*–P9, Senior Developer. Also, *"code review is missing important aspects due to the weakness of coverage in the current predefined checklist of the PL. Enhancing the existing ones and creating new ones to cover other important task types could be beneficial"*–P2, Senior Developer.

Although employing the Spotify model results in a very little standardisation because of having loosely coupled squads that are tightly aligned, it was observed that FinTechOrg tends to be leaner because of the strong culture of cross-pollination besides being transformed from Lean. Some Lean principles adopted by FinTechOrg include Kanban board, continuous process (Lean Thinking), documenting the development process (value stream mapping), and automation whenever possible. A practitioner says:*"The organisation tries to automate the processes whenever is possible by utilising DevOps through continuous integration, delivery, deployment, testing and release"*–P4, Senior Developer. Also, Lean thinking is perceived through the adoption of continuous processes such as continuous backlog grooming, monitoring, prioritisation, feedback, integration, refactoring, verification and testing, deployment, delivery. A practitioner says: *"We have instructions to (1) set up the system for developers, (2) build unit tests, (3) access and deploy, (4) fix release configuration, and (5) do a code implementation and validation using a predefined checklist for the PL"*–P9, Senior Developer.

5.3.2 Collective Decision-Making

Strengthening shared understanding of business objectives: Establishing an organisational strategy, which draws the project's borders and specifies long-term directions, facilitates the identification of scope and specifications. Thus, FinTechOrg emphasises frequently on sharing a common understanding of project scope and specifications, which facilitates decision-making within squads and mitigates possible deficiencies among them. A practitioner says: *"Communicating the project vision and highlighting on the specifications build a common understanding of the project objectives and strengthens our autonomy"*–P10, Product Owner. Further insight about how the organisational strategy is undertaken to strengthen squads' autonomy is provided in Sect. 4.2.2.

Continuously communicating and sharing intentions behind business objectives and providing proper "Definition of Awesome" at different levels of the project facilitates decision-making. Practitioners say:*"We should be familiar with how the GOOD should look like in all areas associated with the provided service!"*–P6, Product Owner and Key Account Manager. Also, *"providing sufficient shared business goals and objectives among the squads aids our people to think in the right direction... Too much shared details might be destructive and wasteful for our resources"*–P7, Product Owner and Key Account Manager.

However, encountering obstacles to decision-making can increase hands-off or results in a waste of resources because of implementing insufficient solutions. Thus, management commitment and support can overcome encountered obstacles and facilitate decision-making. A practitioner says:*"The development of complex features, which require providing dynamic and generic solutions that would be utilised by all customers, might require some support from the management to take the right decision before proceeding in the development"*–P3, Senior Developer.

Strengthening knowledge-based decision-making: Aligning autonomous squads on the product-level might result in a lack of required knowledge and expertise when encountering conflicting priorities, which require working on other squads' tasks. Thus, developers, who are located at another squad and entitled to the desired knowledge

and expertise, should provide some support to those developers who lack the required knowledge. A practitioner says: *"Sometimes, we encounter conflicting priorities between the squads. Hence, some developers from other squads could request some clarifications regarding a specific component, which my squad is entitled to"*–P8, Senior Developer.

FinTechOrg relies on a knowledge-based approach to facilitate decision-making. Instead of relying on domain-independent approach (routine decisions), the organisation values and relies on individuals' knowledge and experience on the product-level and business domain. A practitioner says: *"Taking a decision about which solution to adopt is time-consuming for the developers with insufficient knowledge in this domain and makes it hard to work independently"*–P1, Agile Coach and Architect. Therefore, FinTechOrg focuses on strengthening knowledge sharing to facilitate decision-making and to strengthen squads' autonomy.

Chapters communities facilitate decision-making among autonomous squads. In these Chapters, the individuals are organised based on their skills and competency area. The communities of Chapters align individuals on the skill level. This alignment enables the autonomy of squads and speeds up the decision-making. Practitioners say: *"The members of my Chapter meet regularly and whenever is needed to solve encountered issues or make decisions on how to build things in a standardised way"*–P4, Senior Developer. Also, *"Building cross-functional squads by employing Chapters help in building self-organising squads that work autonomously and effectively"*–P7, Product Owner and Key Account Manager.

5.3.3 Knowledge Sharing

Strengthening the systematic knowledge sharing: The Spotify model employs the communities of Chapters to enable the autonomy of squads by aligning them together. In these chapters, members meet to help in solving problems within their competency areas. However, it is unknown if the Spotify organisation itself emphasises on the continuous sharing of knowledge within chapters. In FinTechOrg, Chapters conduct continuously seminars to share knowledge and experience. A practitioner says: *"sessions*

are conducted to share knowledge and expertise within our chapters... At the end of each session, we plan for the next session and its topic"–P8, Senior Developer.

Also, FinTechOrg employs some practices and processes to systematically share the knowledge: (1) Squad-of-Squads meeting, (2) regular demos and (3) Postmortem Documentation. In Squad-of-Squads meeting, POs communicate encountered challenges, which need to solve, and explain the reasons behind them. The squads are expected to work together to find the best solution and implement it. A practitioner says: *"Emphasising on encountered challenges in our squad-of-squads meetings provides insights on what and why we should improve things. This process stimulates all squads to work towards resolving the challenges"*–P11, Key Account Manager. Conducting regular demo sessions, for newly implemented features, which are of interest for other squads, facilitates knowledge sharing. A practitioner says: *"Conducting regular demos helps all stakeholders to be up-to-date... We will also be able to communicate new features with our customers"*–P13, Key Account Manager. Employing Postmortem Documentation process determines successful and unsuccessful processes on either the process-level or product-level and helps in mitigating future obstacles. A practitioner says: *"Continuously investigating the process and the product after finalising an important milestone or small project facilitates the continuous improvement and knowledge sharing "*–P3, Senior Developer.

Strengthening on-demand knowledge sharing: The organisation utilises informal and on-demand knowledge sharing instead of employing Guilds communities. Since the software development programme of case study project is less than 100 member, the organisation realised that Guilds poses challenges that can be mitigated by employing informal on-demand knowledge sharing process. A practitioner says: *"As the size of our development programme is less than Spotify's one, we replace the Guilds with a call for meetings through Slack to invite those who are interested in the subject under discussion"*–P7, Product Owner and Key Account Manager.

On the contrary to the Spotify organisation, it was observed that FinTechOrg utilises a *limited fail-friendly culture* since it provides a FinTech service. A practitioner says: *"Release failure could cause a loose of money and impact our reputation negatively"*–

P5, Product Owner. To eliminate possible release failures, the organisation employs a limited fail-friendly culture. Since failures can be encountered during the pilot launch of newly implemented features, the responsible squad shall decide whether to switch off newly release features or to roll back the whole release. A practitioner says: *"In case of encountering a bug, we either switch off newly implemented code if possible or rollback the release... Then, we fix the code and increase the test coverage"*–P8, Senior Developer. These encountered failures are embraced to highlight learning lessons and to improve the project and the process. A practitioner says: *"Sharing the encountered issues continuously in our squad-of-squads meetings provides the ability to take actions to improve both product and the process"*–P10, Product Owner.

The organisation utilises peer code review, either within the same squad or on the squads level, to ensure the successful development of FinTech software service. Encountering contradicted priorities between the squads sometimes causes the other squad, which lacks the expertise and knowledge on the targeted product-level, to take over and handle such work items. Thus, a peer code review on the squad level is required to secure the development. A practitioner says: *"Sometimes, we take over other squads' tasks since they do not have enough resources... Afterwards, code review and inspection is employed on the squad level to ensure the continuity of handing in successful deliverables"*–P9, Senior Developer.

5.3.4 Inter-team Coordination

The Engineering Culture of Spotify does not employ inter-team coordination practices and processes. This ignorance is because of the needs for speeding up the process by having autonomous squads instead of relying on managers who should coordinate among the squads. However, the existence of contradicted situations demands inter-team coordination. It was observed that FinTechOrg emphasises on having sufficient inter-team coordination to prevent excessive processes that could waste resources and hence impact negatively the autonomy of squads. A practitioner says: *"We do not want to have too much coordination and alignment that might mitigate squads autonomy... We employ just enough coordination to overcome the challenges"*–P1, Agile Coach and Architect.

Strengthening formal coordination: Formal inter-team coordination is mostly depicted through two main routine meetings: Squad-of-Squads meetings and POs meetings. In Squad-of-Squads meeting, POs provide brief information about what their teams are determined to do by sharing their intents and obstacles (if there were any) instead of reporting the work progress status. A practitioner says: *"I explain briefly in our Squad-of-Squads meeting what my squad is intended to do in the upcoming period and the encountered challenges"*–P7, Product Owner and Key Account Manager.

In POs meeting, a synchronisation meeting is conducted to ensure (1) the alignment between squads' missions, (2) the ownership of the product itself, and (3) the alignment for the overall road map of the organisation. Practitioners say: *"Our POs always try to emphasise on the ownership of the provided software service since the customers often try to push product development in their own ways"*–P2, Senior Developer. Also, *"We have regular meetings between the POs to tackle important tasks in the upcoming iteration(s). Also, POs meet up to discuss and prioritise intersected tasks to aligned between the squads"*–P4, Senior Developer. Moreover, *"In our POs meetings, we match and distribute the backlog items based on squads missions to align squads over the roadmap"*–P5, Product Owner.

Strengthening informal coordination: Informal inter-team coordination was observed in the employment of non-routine and on-demand meetings. This informal inter-team coordination is utilised to resolve conflicting priorities between the squads and to control the outcome of intercorrelated tasks.

On-demand informal inter-team coordination is conducted between the POs to resolve the conflicting priorities between the squads. A practitioner says: *"We encounter situations where other squads, who is responsible for a specific part of the system, are not able to handle specific work items. Therefore, the POs arrange between their squads whenever is needed"*–P3, Senior Developer.

Non-routine informal inter-team coordination process is employed between involved squads' members to control the outcome of some complex tasks. A practitioner says: *"Some work items require the involvement of multiple stakeholders from multiple squads to decide on which solution to adopt since the uncertainty is high and due to the needs*

for covering the full lifecycle of the project"–P5, Product Owner. Otherwise, time and resources could be wasted.

5.3.5 Mission-Based Planning

The squads respond to customers' needs at different velocities based on their missions and their scaled agile methods. The findings show that the planning activity is affected by the nature of squads' missions. While innovation based missions neglect estimation and embrace Lean Startup, the plan based missions embrace automation, standardisation, and estimation. This section ignores "Strengthening PL" in which plan-based missions embrace automation, standardisation, and estimation since it is described in detail in Sect. 6.4.3 through *strengthening product-line development and automation*.

Strengthening innovation: The Spotify model encourages the utilisation of Lean Startup to promote innovation, likewise FinTechOrg. Tasks of maintenance nature (i.e., adaptive or perfective) and/or of newly requested features are characterised with a high degree of uncertainty. A practitioner says: *"Developing new features and adapting or improving already existed ones impose challenges due to the high level of uncertainty... providing dynamic and generic solutions increase the complexity"*–P9, Senior Developer. Such tasks require innovation to provide customers with the desired business values. Hence, those squads tackling such tasks have missions that embrace Lean Startup principles. A practitioner says: *"We have hybrid process based on Lean Startup and Kanban"*–P10, Product Owner.

Software development estimation is considered as a waste for those squads with innovation missions. Practitioners say: *"We sacrifice the predictability of delivery for the sake of providing our customers with valuable features"*–P6, Product Owner and Key Account Manager. However,*"customers request sometimes an estimation before starting the development... We provide a rough estimation and keep them involved in the process to revise the plan accordingly"*–P12, Product Owner.

5.3.6 Release/Delivery Strategy

Speed up the release delivery: The organisation employs a decoupled architecture as suggested by the Spotify model to make a release train for each part of the software. A practitioner says: *"We utilise a decoupled architecture to (1) facilitate the alignment of squads on the product-level, (2) mitigate possible dependencies between squads, and (3) prevent impacting the whole system when a mistake is introduced"*–P9, Senior Developer. Since the squads are aligned on the product-level and a decoupled architecture is utilises, each squad can have a different release train, as illustrated in Fig. 5.4. A practitioner says: *"The squads are aligned on the product-level since a decoupled architecture is employed... each part of the software has its release train"*–P4, Senior Developer. This employed strategy would, in turn, strengthen squads autonomy and facilitate the delivery of frequent small releases, which consequently will speed up the process of delivery.

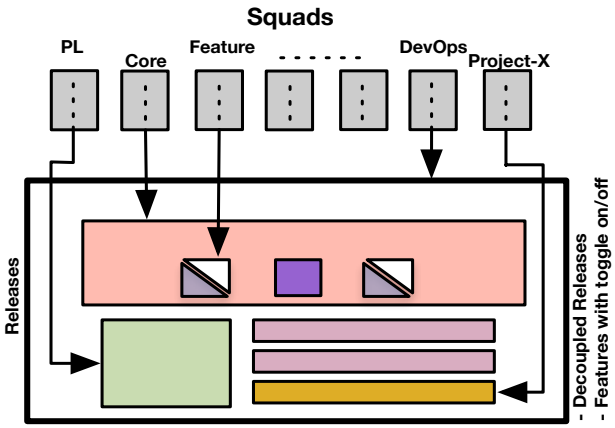


Figure 5.4: The nature of squads missions

Strengthening the accountability for release delivery: Despite allowing the release of unfinished code as hidden by the Spotify model, it was observed that FinTechOrg does not do that in-spite of providing new features with a toggle on/off switch. Forbidding releasing unfinished code aims to ensure (1) having clean code base that eliminates possible inconsistencies between the squads since collective code ownership is adopted, and (2) having stable working software features at the production platform since the organisation provides a FinTech service. A practitioner says: *"It is crucial to have*

a clean code base that only has stable working features as the code is shared by all squads"–P1, Agile Coach and Architect.

It is unknown if the Spotify model utilises on-demand releases in case of missing a release train. FinTechOrg *"employs DevOps to automate the process of release delivery whenever is needed"*–P4, Agile Coach and Architect. Also, the organisation releases on-demand in case of encountering a situation where a squad missed a release train, and a customer requested a release delivery. A practitioner says: *"If we missed a release train this week, we either wait for the next train or we can deliver the finished work whenever is demanded by a customer"*–P7, Product Owner and Key Account Manager.

To strengthen the release strategy, the organisation provides backwards compatible releases in order to prevent any deviation in the behaviour of the software service from the intended one in the old release and to strengthen squads' autonomy. A practitioner says: *"We always make sure that old features, components, and integrated APIs as well as their old configuration files working as expected... This is to satisfy customers' needs and to prevent possible conflicts of interests between the squads"*–P4, Senior Developer. Also, having backwards compatible releases is considered powerful to facilitate the process of rolling back a release in case of encountering problems. A practitioner says: *"A key rule of thumb to always follow is to provide backwards compatible releases. This process, in turn, facilitates the process of rolling back when facing problems"*–P9, Senior Developer.

The organisation mitigates possible release failures through the utilisation of Limited Blast Radius technique. This technique facilitates the verification of software releases over a limited number of customers and a limited number of end-users instead of rolling it out directly for all customers and end-users. A practitioner says: *"We utilise a limited blast radius process and monitor new releases to verify the software behaviour for few customers with limited number of end-users... In case of encountering a bug, we either switch off newly implemented code if possible or rollback the release... If things were successful, we roll it out to the rest of customers and end-users"*–P8, Senior Developer.

5.4 Summary

This chapter addresses the research question (RQ1.2): *How do agile practitioners, from a FinTech organisation, achieve and sustain the alignment of autonomous squads?* A longitudinal embedded case study was conducted to answer this question. The data were collected through observing 225 ceremonies over 21 months, conducting 14 semi-structured open-ended interviews, and accessing different sorts of artefacts. The collected data was analysed using an approach informed by the Grounded Theory method.

The analysis, in this chapter, has identified factors that influence the the alignment of autonomous squads. Investigating Spotify Tailoring, from a squad alignment perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. This investigation facilitated the identification of the FinTech's Spotify model, from squads alignment perspective. In result, *31* tailored practices and attributes, which facilitate the alignment of autonomous squads, were identified. Some identified practices and attributes are previously introduced by the Spotify model, such as the two-dimensional structure, Chapter based decision-making, and peer code review on the squad level. Also, some identified practices and attributes were modified, such as neglecting Guilds and Tribes communities, and not releasing unfinished code. Moreover, other new practices and attributes were introduced, such as covering the full value chain and whole lifecycle and management commitment.

Also, the analysis identified influential factors on aligning autonomous squads. These influential factors are; (1) adaptive structure, (2) collective code ownership, (3) collective decision-making, (4) knowledge sharing, (5) inter-team coordination, (6) mission-based planning, and (7) delivery strategy. Each identified factor is supported by a set of practices and attributes that strengthens the alignment of autonomous squads. The identified influential factors and their related practices can aid agile practitioners in aligning autonomous squads.

Chapter 6

Spotify Tailoring for B2B Product Development

6.1 Introduction

This chapter addresses the research question (RQ1.3): *How do agile practitioners, from a FinTech organisation, tailor the Spotify model for B2B product development?* A longitudinal embedded case study was conducted, as described in Sect. 3.4.

The findings described in this chapter reveal 2 aspects related to product development. Firstly, the impact of product development on squads autonomy and alignment was identified. Secondly, performing B2B product development by tailoring the Spotify model was identified. The investigation of Spotify Tailoring, from a B2B perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. This investigation facilitated the identification of the FinTech's Spotify model, from a B2B product development perspective. Also, the investigation of Spotify tailoring has identified influential factors on Spotify tailoring for B2B product development. Each identified factor is supported by a set of practices and attributes that facilitate B2B product development. The identified influential factors and their related practices can

aid agile practitioners in performing B2B product development by tailoring the Spotify model. Fig. 6.1, which is depicted in the top side of the whole model illustrated in Fig. 7.2, illustrates briefly the emergence of the project strategy and how the organisation interacts with the customers.

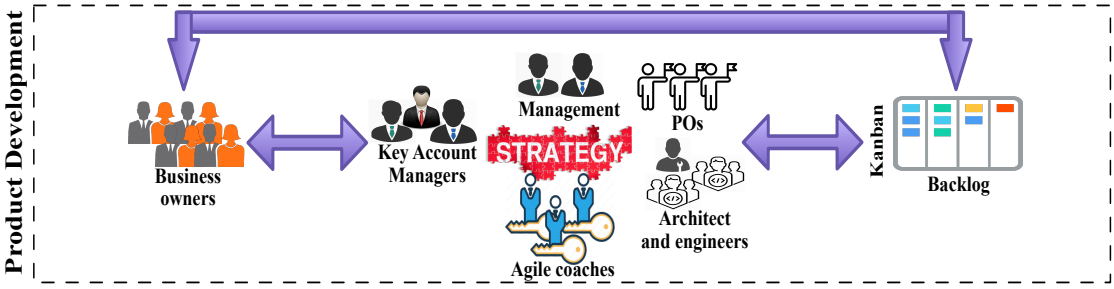


Figure 6.1: B2B product development

This chapter is an expansion of a peer-reviewed research paper [95].

6.2 The impact of product development on squads autonomy and alignment

The researcher has observed that the project strategy is shaped by the product steering committee, which consists of a management team, Key Account Managers, POs, Agile coaches, and architects. Hence, software product development is influenced by the organisational strategy and its road map. This product development impacts squads autonomy and alignment.

Product development impacts squads autonomy through identifying the strategy of squads missions. The product steering committee draws the strategy of squads missions. This identified strategy is used to facilitate tailoring agile practices for each squad to achieve its mission ultimately. Hence, the identified strategy facilitates the process of establishing and building autonomous squads. Practitioners say: *"Our product steering committee draws the strategy for each squad's mission to enable self-governance"*–P7, Product Owner and Key Account Manager. Also, *"identifying the strategy of squads missions*

helps us in tailoring agile practices for each squad based on its own mission"–P1, Agile Coach.

Product development impacts squads alignment. The product steering commite, through POs, aligns product backlog items among autonomous squads. POs organise the product backlog by prioritising the tasks and tickets and distribute them among the squads according to their missions. A practitioner says:*"We (POs) meet up regularly to prioritise and align backlog tasks among squads"*–P10, Product Owner.

6.3 Spotify Tailoring for B2B Product Development – Practices and Attributes

This research study identifies practices and attributes that facilitate performing B2B product development in an offshore outsourced FinTech project of large-scale. The identified practices and attributes are tagged by α, β, and γin Fig. 6.2. "α" represents practices and attributes that are previously introduced by the Spotify model. "β" represents Spotify's practices and attributes that were modified by FinTechOrg. "γ" represents newly introduced practices and attributes by FinTechOrg.

In result, *44* tailored practices and attributes, from a B2B perspective, were identified. These practices and attributes, which represent the emerged Codes in the Grounded Theory as presented in Fig. 6.2, are classified into three main types, as follows:

1. *Already exist: 10* practices and attributes, which are enumerated in Table 6.1, were followed by FinTechOrg as previously introduced by the Spotify model. These practices and attributes are tagged by α in Fig. 6.2.

Table 6.1: Already existing practices and attributes in the Spotify model, from a B2B perspective

Already existing practices in the Spotify model
Definition of Awesome
Limited blast radius
No estimation process for innovation based missions
Employing Lean Startup
Features with toggle switch
Squad autonomy
Horizontal alignment
Decoupled architecture
Decoupled releases
Squad or mission based frequent and small releases

2. *Modified: 6* adapted Spotify practices and attributes by FinTechOrg, which are enumerated in Table 6.2. These practices and attributes are tagged by β in Fig. 6.2.

Table 6.2: Adapted Spotify practices and attributes by FinTechOrg, from a B2B perspective

Adapted practices and attributes
Clear vision and roadmap
Defined project scope and specifications
Ownership of the product
Aligning the customers over road map and strategy
Postmortem Documentation on B2B level
Limited fail-friendly culture

3. *New: 28* new practices and attributes, which are enumerated in Table 6.3, were introduced by FinTechOrg. These practices and attributes are tagged by γ in Fig. 6.2.

Table 6.3: New practices and attributes introduced by FinTechOrg, from a B2B perspective

New practices and attributes
Defined milestones
Utilising globally accessible project management and issue tracker tools
Well defined and documented process for B2B product development
Involvement of relevant stakeholders
Routine and on-demand meetings
Risk and opportunity identification
Quick handling of continuous changes in customers' business
Identification of key contact (1st line) in the both sides of B2B
Support squad works as the 2ed line of contact
Knowing customers' business domain
Having good experience
Knowledge based decision-making
Developers hold Proxy-PO role
Transparency and mutual respect
Continuous sharing of intent
On-demand meetings to control the outcome of tasks
Routine meetings to draw path - Management support
Emphasising on the privacy and security of provided software system
Financial and legal stability
Proactive management style
Eliminate surprises for customers
Promise less but deliver more
Involve the customers
Utilising a product-line
Task standardisation for the product-line
Adopting Lean Thinking in the product-line squad
Adopting Scrumban in the product-line squad
Up-front estimation for the tasks of the product-line

6.4 Influential Factors on B2B Product Development

The analysis of the collected data has grounded a synergy between the following categories (i.e., factors), which are depicted in Fig. 6.2, and facilitating B2B product

development. Fig. 6.2 depicts the emerged categories along to their related concepts and codes. This figure is built by following the model illustrated in Fig. 3.3, which presents the emerging levels of abstraction when using the Grounded Theory. *Spotify Tailoring* is presented in Table 6.4 to indicate the coverage of the adopted practices and attributes, which are mentioned in the previous section (Sect. 6.3), by the Spotify model and FinTechOrg. The influential factors, which represent the emerged Categories and Concepts of the grounded theory were used as a pattern to present Spotify Tailoring from a B2B perspective. Moreover, Table 6.4 clarifies the extent of which the Spotify model, in terms of product development, has been scaled in the organisation. The following sections present the influential factors on B2B FinTech product development.

Table 6.4: Influential factors on B2B product development

Factors	**Concepts**	**Spotify**	**Case Study**
Project visibility	Shared understanding of the business objectives	Yes	Yes
	Globally accessible project management tools	No	Yes
	Embracing offshore product development processes	No	Yes
Employed interactions	Collaboration and coordination	No	Yes
	Communication of requirements and organisation's capabilities	No	Yes
	Facilitating decision-making	Yes	Yes
B2B relationships	Accountability for the provided software service	Yes	Yes
	Expectation management	Unknown	Yes
	Relationships among stakeholders	No	Yes
Response time	Proper delivery strategy	Yes	Yes
	Facilitating innovation by employing Lean Startup	Yes	Yes
	PL development and automation increase predictability	No	Yes

Yes: covered, ≈**Yes**: partially covered as depicted in Fig. 6.2, **No**: not covered, **Unknown**: no evidence

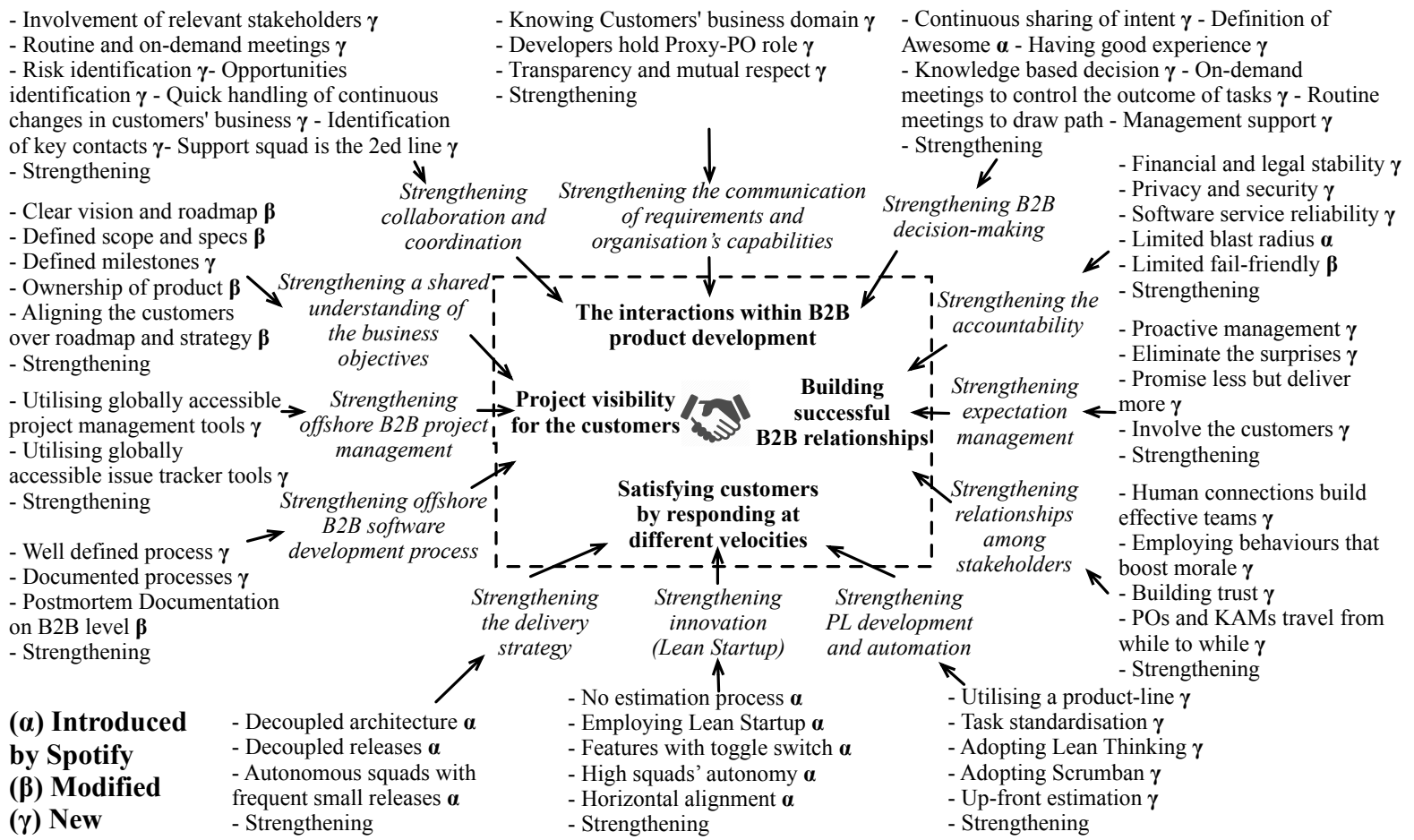


Figure 6.2: Emergence of the influential factors on Spotify Tailoring, from a B2B product development perspective, which is adapted from [95]. The **categories** are in bold, the *concepts* are in italic, and the codes are in plain text

The following subsections describe the emerged categories and their related concepts and codes, which are depicted in Fig. 6.2. These categories represent influential factors on performing B2B product development.

6.4.1 Project Visibility for the Customers

Strengthening a shared understanding of the business objectives: Establishing clear project vision, by defining the scope and a set of specifications, provides customers with directions to avoid confusion during the development process. Practitioners say:*"project vision is communicated frequently, but it changes quite often. This is mainly due to market demands since it is volatile."*–P10, Product Owner. Also, *"frequently communicating clear targets and establishing milestones based on the defined specifications is considered important for both of the squads and the customers"*–P7, Product Owner and Key Account Manager. To be able to maintain a shared project vision and specifications, POs have a regular meeting to ensure of (1) the alignment of the product strategy and the overall roadmap of the organisation, and (2) the ownership of the provided software service itself. A practitioner says: *"We have a weekly meetings between the POs to prevent the deviation from product's main purpose... the provided features should not only cover the needs of a specific customer, but they should also be usable by all customers within a specific domain"*–P12, Product Owner.

Strengthening offshore B2B project management: FinTechOrg utilises globally accessible project management tools (e.g., Rally) and issue tracker tools (e.g., Jira) to increase the visibility of this FinTech software project within this B2B environment. Providing customers with shared online Kanban boards facilitates project management through the utilisation of this mechanism, which has a low coordination overhead. A practitioner says: *"To align B2B product development, we provide our customers with shared Kanban boards to facilitate the planning activities as customers can observe the progress and take decisions accordingly."*–P11, Key Account Manager. Since the organisation has a strong culture of cross-pollination, all squads employ Kanban to utilise a standardised operational process that is controlled by employing the definition of "DONE". A practitioner says:*"The definition of "DONE" ensures tasks' completeness,*

rules the overall process flow and satisfies customers' needs"–P7, Product Owner and Key Account Manager. Hence, the squads respond to customers' needs in a highly transparent and disciplined manner.

Strengthening offshore B2B software development process: FinTechOrg embraces well-defined and documented processes to improve project and process visibility in this B2B environment since the customers are scattered around the globe. A practitioner says: *"We share, continuously, with our customers feature instructions, release notes, and highlight on software development processes to facilitate the business and software development process"*–P10, Product Owner. Also, Postmortem Documentation process is utilised by FinTechOrg to document project-related aspects continuously and to improve the development process. Therefore, *"at the end of each small project, we get customer's feedback to improve the product and the process if needed"*-P11, Key Account Manager.

6.4.2 The Interactions within B2B Product Development

Strengthening the collaboration and the coordination: Product development in the case study project demands sufficient collaboration and coordination between the two sides of B2B. Regular and on-demand involvement of multiple stakeholders from both sides of B2B is required. This is to handle more complex issues, which are characterised with a high level of uncertainty. A practitioner says: *"Key players from both parties meet up on a weekly basis to discuss the progress over an online Kanban board... we also meet up whenever is needed."*–P11, Key Account Manager. Involving both sides of B2B in the process of product development keeps all sides informed, enables the identification of potential opportunities that both sides might invest in, and enables the identification of possible risks that both sides might overlook. A practitioner says: *"The involvement of key stakeholders from both sides is considered crucial. Otherwise, we may realise that things should have been handled differently. Thus, our priorities come into focus or changing"*–P7, Product Owner and Key Account Manager. This continuous involvement helps in tackling the encountered obstacles, making decisions, and highlighting on priorities.

Identifying key contact persons in the two sides of the B2B environment, for this case study project, strengthens the interaction in business and product development. Also, this identification facilitates a continuous communication flow between the two sides. Practitioners say: *"We (i.e., POs and KAMs) do highlight on external communication hierarchy, internal structure, and squads' missions to make sure that all parties are on the same page"*–P10, Product Owner. Also, *"our support squad continuously helping the customers, in a low Resolution-Time, with service configuration management, issues investigation, requirements engineering, and continuously giving feedback to the POs and KAMs"*–P14, Support Manager. It was observed that this support squad represents the second line of contact for the customers to strengthen collaboration and coordination.

Strengthening the communication of requirements and organisation's capabilities: In this B2B environment, the product development of the case study project is tightly linked with (1) the continuous communication of requirements and (2) the continuous communication of the organisation's capabilities to the customers. Misunderstanding of the project requirements or organisation's capabilities within a specific period puts even simple implementations at serious risk.

Understanding the business domain of the customers in this B2B environment facilitates precise interpretation and elicitation of the requirements. *"When customers request a new feature as much information as possible should be provided by the customer to speed up the support and the development processes"*–P14, Support Manager. Since newly requested features are usually characterised with high-level of uncertainty, developers take the role of Proxy-POs to extract the requirements and to get the requested featured matured enough before starting software development. A practitioner says: *"Usually, I get in touch with our customers to extract the requirements due to the vagueness of newly requested features"*–P9, Senior Developer. Employing an iterative way of development and getting continuous feedback mitigates risks. A practitioner says: *"We use live meetings, Slack channels, or emails to extract the requirements iteratively and to approve the changes... The requirements are listed in the work items of Kanban"*–P5, Product Owner.

Both sides (FinTechOrg and customers) of this B2B environment have different interests,

but they overlap. The customers utilise the provided software service to perform a business functionality that is outside their main business domain. However, the provided FinTech software project, by FinTechOrg, handles a crucial part of customers' business. A practitioner says: *"A successful product development relies on fulfilling customers' needs and maintaining respect for what we need to accomplish"*–P10, Product Owner. FinTechOrg has developed a culture based on mutual respect and transparency with the customers. A practitioner says: *"continuously communicating what capabilities (time and resources) we can provide within the upcoming period"*–P11, Key Account Manager. This culture helps the customers in realising the capabilities that can be provided by FinTechOrg in the upcoming period to streamline the process of product development at the customers' side.

Strengthening B2B decision-making: FinTechOrg emphasises a continuous sharing of intentions at different levels of product development to facilitate decision-making. A practitioner says:*"As we are familiar with the intentions behind each component or feature, we can make the right decisions quickly"*–P6, Product Owner and Key Account Manager. Also, FinTechOrg utilises the*"Definition of Awesome"* in each area of the provided software product, which in turn facilitates decision-making for POs and Key Account Manager. A practitioner says:*"Possessing good knowledge about how the GOOD should look like in all areas improves our product development"*–P7, Product Owner and Key Account Manager.

Decision-making in FinTechOrg is mostly shifted from a domain-based approach, which is based on routine decisions, to a knowledge-based approach, which is based on expertise. A practitioner says: *"Our POs' squad have long experience and knowledge in this industry, which in turn facilitates our work"*–P12, Product Owner. Nevertheless, on-demand decision-making and regular meetings are conducted between the POs to control the outcome of some tasks or to ensure the ownership of the project. A practitioner says: *"Whenever encountered tasks with high complexity and uncertainty, we investigate the possible outcome such task... Also, we (POs) meet up regularly to ensure the ownership of our project as customers are enthusiastic about using our software service in their own way"*–P11, Product Owner.

Sustainable management support facilitates collective decision-making and speeds up product development. POs, in FinTechOrg, encounter situations in which the management support is considered necessary despite the utilisation of regular and on-demand meetings between the POs. A practitioner says: *"We get back to the management in case of encountering show stoppers, identifying possible risks... and when finding possible opportunities to invest in"*–P10, Product Owner. Hence, the management's support strengthens decision-making by eliminating hands-off, resolving potential risks, and speeding up the process.

6.4.3 Building Successful B2B Relationships

Strengthening the accountability: Customers in this case study rely on the organisation's accountability to provide reliable, stable, and secure financial software service, which impacts the Return Of Investment (ROI). This reliance is because the case study project offers customers' end-users with a wide range of payment solutions. A practitioner says: *"if things go wrong with the provided software service, it will negatively affect customers' business, reputation, and revenues"*–P11, Key Account Manager. This FinTech project is mission-critical, which influences customers' projects since the project plays a vital role in the lifecycle of customers' projects. Practitioners say: *"end-users assume that if a specific part of a customer's product is bad, then the whole product is bad too"*–P13, Key Account Manager. Also, *"the stability of the provided service is also about the financial and legal stability of the provided software"*–P1, Agile Coach and Architect.

FinTechOrg emphasises on providing 24/7 stable software service constantly. Hence, FinTechOrg emphasises on the employed practices to conform with the case study project and to serve customers in their competitive advantage. For instance, instead of rolling out newly implemented features to all customers at once, a *"Limited Blast Radius technique is utilised to perform experiments on a limited number of end-users"*–P8, Senior Developer. Consequently, the responsible squad decides whether to gradually roll out the new feature or roll it back based on the monitoring results. Also, since a configuration-driven development approach is employed in this case study project to control the behaviour of

the software system at the execution time, the organisation *"usually makes backwards compatible releases to be able to roll back in case of encountering issues"*–P2, Senior Developer.

FinTechOrg employs a limited fail-friendly culture because of providing a FinTech software service. A practitioner says: *"As we provide financial software services, failure is not tolerated since it affects our reputation directly"*–P1, Agile Coach and Architect. However, failures are inevitable, sometimes, during the pilot launch of a newly implemented code, which aims to improve and verify features' behaviours. These failures are utilised to learn and improve the process and the product. A practitioner says: *"In our squad-of-squads weekly meeting, we share the reasons behind encountered release issues to improve the product and the process if needed"*–P12, Product Owner.

Strengthening expectations management: Managing the expectations of the customers proactively builds strong B2B relationships and satisfaction. A practitioner says: *"a deeper understanding of the customers, at company and business domain levels, is considered important to provide customers with better solutions that would strengthen their competitive advantage"*–P13, Key Account Manager. FinTechOrg works on eliminating possible surprises when working on tasks with a high level of uncertainty and complexity. For instance, *"announcing the initiation of implementation for a task brings customers' excitement and builds expectations. Not being able to deliver on time causes dissatisfaction and a loose of confidence in our ability to produce in the future."*–P7, Key Account Manager. Also, *"we (i.e., POs and KAMs) tend always promise less but try to deliver even more*–P5, Product Owner. To mitigate possible disappointment, POs and KAMs continuously involve the customers in the process, which in turn eliminates possible misunderstandings. A practitioner says: *"When meeting customers, we try to avoid possible pitfalls by having a conversation in which both sides openly discuss what is expected from the other side to prevent possible misunderstandings"*–P7, Product Owner and Key Account Manager.

Strengthening relationships among stakeholders: B2B product development relies on the relationships among stakeholders to overcome increasingly complex challenges. *"Maintaining proper communication and connections reinforces the notion of that sides*

of B2B are all in the same boat"–P10, Product Owner. Thus, the two sides together are truly one team working towards the same goal. Also, *"the communication includes things like congratulating the other part for an accomplishment, such as reaching a milestone"*–P11, KAM. Such behaviours boost morale between both sides and build strong relationships. Since the organisation has developed an environment based on transparency and mutual respect with the customers, trust is built by time. However, *"to gain the trust of new customers, we tend to travel sometimes to meet customers, carry out on-site training sessions, show the customers our track record, state-of-the-art technology, existing expertise and also showing the willing to understand their real needs"*–P7, PO and KAM.

6.4.4 Satisfying customers by responding at different velocities

FinTechOrg responds to customers' needs at different velocities based on the employed process in each squad, which is based on the mission of the squad. While some squads missions value innovation more than plan fulfilment, other squads value plan fulfilment more than innovation. In this section, *strengthening the delivery strategy* and *strengthening innovation* are ignored since they are covered by the Spotify model [57], and in Sect. 5.3.5–5.3.6.

Strengthening product-line development and automation: The utilisation of a software product-line provokes task standardisation to facilitate the process of integrating the case study project into external payment APIs (sub-systems). Since this project manages autonomous software systems, *"the project utilises a PL architecture, which facilitates the process of integrating the project into external sub-systems"*–P9, Senior Developer. Eliminating waste using Lean Thinking is harnessed through the employment of predefined checklists. A practitioner says: *"checklists are utilised to facilitate requirement extraction, code review, planning, estimation, documentation, knowledge sharing, etc."*–P8, Senior Developer. These checklists help FinTechOrg in automating the process for those squads working on the PL and help in speeding up the development. Since PL work items are characterised with a low degree of uncertainty, an up-front estimation process is considered beneficial for the customers. PL based squads do utilise a tailored

process based on Lean and Scrumban. A practitioner says: *"We employ some processes from Lean and Scrumban... we use bucket size, on-demand planning techniques, and average lead/cycle time"*–P5, Product Owner. Hence, POs realised more confidence in promising customers with predictable delivery deadlines.

6.5 Summary

This chapter has addressed the research question (RQ1.3): *How do agile practitioners, from a FinTech organisation, tailor the Spotify model for B2B product development?* A longitudinal embedded case study was conducted. The data were collected through observing 225 ceremonies over 21 months, conducting 14 semi-structured open-ended interviews, and accessing different sorts of artefacts. The collected data was analysed using an approach informed by the Grounded Theory method.

The analysis, in this chapter, revealed 2 aspects related to product development. Firstly, The impact of product development on squads autonomy and alignment was identified. Since product development is influenced by the organisational strategy and its road map in the industry, the product steering committee is responsible for drawing the strategy of squads missions and aligns product backlog items among autonomous squads. Identifying squads strategy, in turn, enables squads autonomy and align them to common product development objectives.

Secondly, performing B2B product development by tailoring the Spotify model was identified. The investigation of Spotify Tailoring, from a B2B perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. This investigation facilitated the identification of the FinTech's Spotify model, from a B2B product development perspective. In result, *44* tailored practices and attributes, which facilitate B2B product development, were identified. Some identified practices and attributes were previously introduced by the Spotify model, such as the definition of awesome, limited blast radius, and decoupled architecture. Also, some identified practices and attributes were modified, such as the identification of clear vision and

roadmap, postmortem documentation on a B2B level, and limited fail-friendly culture. Besides, other new practices and attributes were introduced to the Spotify model, such as using globally accessible project management and issue tracking tools, developing good knowledge about customers' business domains, and proactive management style.

Also, the investigation of Spotify tailoring has identified influential factors on Spotify tailoring for B2B product development. These factors are (1) project visibility, (2) employed interactions, (3) relationships, and (4) response time. Each identified factor is supported by a set of practices and attributes that facilitate B2B product development. The identified influential factors and their related practices can aid agile practitioners in performing B2B product development by tailoring the Spotify model.

Chapter 7

Heterogeneous Tailoring Approach

7.1 Introduction

This chapter addresses the research question (RQ2): *What is the approach taken to agile tailoring, when using the Spotify model?* To answer this research question, a longitudinal embedded case study was conducted, as described in Sect. 3.4.

The findings described in this chapter reveal a novel approach to agile tailoring, called *"Heterogeneous Tailoring"*. This approach was observed by looking at agile tailoring through the lens of Spotify Tailoring. This chapter identifies the characteristics, benefits, and challenges of the Heterogeneous Tailoring approach. The Heterogeneous Tailoring approach is illustrated briefly in Fig. 7.1.

This tailoring approach comprises 3 key features, which are presented in Chapter 4, Chapter 5, and Chapter 6. Chapter 4, which describes building and establishing autonomous squads and how each autonomous squad is empowered to select and tailor its development method, is illustrated in the bottom part of Fig. 7.1. Chapter 5, which describes the alignment of autonomous squads, is illustrated in the middle part of Fig. 7.1. Chapter 6, which describes performing B2B product development in FinTech's Spotify model and how the product steering committee draws the strategy of squads' missions

and aligns the product backlog among squads, is illustrated in the top part of Fig. 7.1. In Fig. 7.1, the used practices and processes by each autonomous squad and for aligning all squads are collected into a repository that is managed by the Agile Coaches.

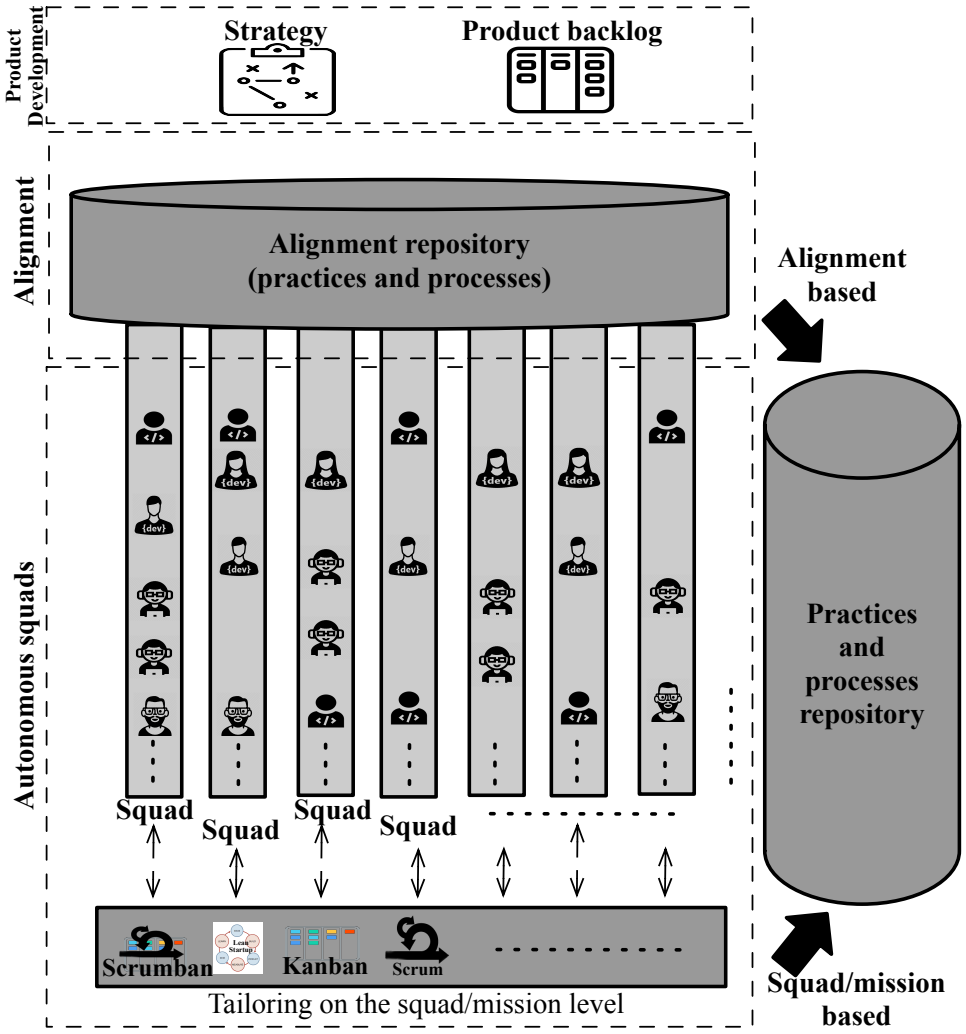


Figure 7.1: A Heterogeneous Tailoring approach [98]

This chapter is an expansion of a peer-reviewed research paper [98].

7.2 Characteristics of the Heterogeneous Tailoring Approach

The Heterogeneous Tailoring approach is characterised by three key features. Firstly, each autonomous squad is empowered to select and tailor its development method. This key feature is described in detail in Chapter 4. Secondly, each squad is aligned with other squads and to common product development goals and objectives. This key feature is described in detail in Chapter 5. Thirdly, product steering committee draws the strategy of squads' missions and aligns the product backlog among squads. This key feature is described in detail in Chapter 6.

This Heterogeneous Tailoring approach is illustrated in detail in Fig. 7.2, which depicts the three main key features of this tailoring approach. The top side of the figure represents product development and how it impacts the rest of the features of this approach. The middle side of the figure depicts the alignment of the heterogeneous autonomous squads. The bottom side of the figure illustrates the heterogeneity and autonomy of the squads.

The product steering committee draws the strategy of squads missions. Based on the drawn strategy of squads missions, each autonomous squad can select and tailor its development method while getting proper support from the Agile Coach. The identification of squads mission, in turn, facilitates the process of creating autonomous squads. The squads should be aligned with each other and to common product development objectives to enable and strengthen their autonomy. Also, the product steering committee aligns product backlog items among the autonomous squads based on their missions.

This Heterogeneous Tailoring approach is mainly concerned with creating autonomous yet aligned squads. The alignment of squads works as an enabler for squads autonomy. Also, product development enables both of the alignment and the autonomy of squads.

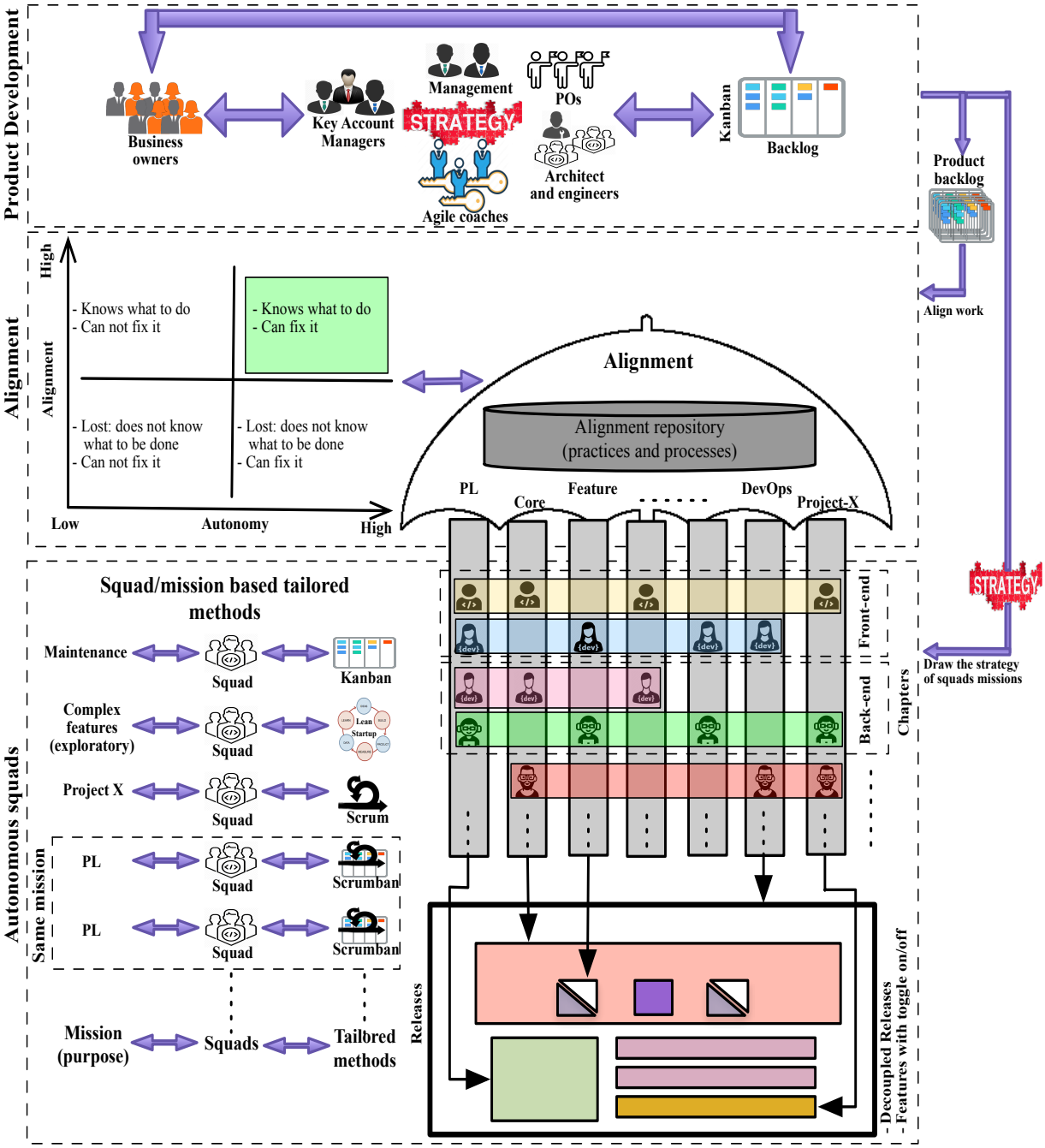


Figure 7.2: The Heterogeneous Tailoring Approach

7.3 Benefits of the Heterogeneous Tailoring Approach

The practitioners, in FinTechOrg, reported benefits of this Heterogeneous Tailoring approach. These benefits include: (1) mitigating challenges of utilising agile practices across squads that do not fit the needs of each squad, (2) improved creativity of some squads and productivity for others, (3) and mitigated risks of divergence from shared development objectives by employing alignment practices. These reported benefits, in turn, facilitate strengthening the autonomy of squads.

The Heterogeneous Tailoring approach mitigates challenges of utilising agile practices across squads that do not fit the needs of each squad. Each squad, in FinTechOrg, has the freedom to independently tailor agile methods based on its mission while having required support from agile coaches. Practitioners say that *"our squad has the freedom to adapt agile practices to fit our squad's needs... This way, we have a development process that suits our squad that consequently speeds up the development process"*–P8, Senior Developer. Also, *"tailoring squads' processes based on their missions increases the ability to incorporate squad or mission-based processes... This tailoring mitigates the needs of handling general practices and processes that could be challenging for some squads and their missions"*–P1, Agile Coach and Architect.

Also, the practitioners in FinTechOrg reported benefits of improved creativity for some squads and the productivity for others by the ability to evolve their agile practices based on their missions. Practitioners say: *"employing agile practices that suites my squad helps us in improving our productivity and creativity since we adapt our practices to be compatible with our mission and needs... While some squads value productivity, such those working on the product-line, other squads value more the creativity"*– P12, Product Owner. Also, *"adapting our squad's practices to conform with our mission minimises the employment of inapplicable general practices... Customising our own practices speeds up our development process and facilitates accomplishing our own mission"*–P3, Senior Developer. Hence, the employment of squad or mission-based agile tailoring preserves the autonomy of squads and improves the creativity for some squads and increases productivity for others.

In addition, the practitioners reported mitigated risks of divergence from shared development objectives through alignment practices. The Spotify model creates alignment on the product-level to facilitate the creation of expertise in specific areas. Also, the Spotify model creates alignment by employing an adaptive structure, which is based on a matrix of two dimensions, and by creating communities around the structure. However, FinTechOrg tailors the Spotify model to fit the needs of their projects and contexts by expanding the alignment of these heterogeneous autonomous squads to include other factors. These factors, which are described in detail in Chapter 5, include; (1) adaptive structure, (2) collective code ownership, (3) decision-making, (4) knowledge sharing, (5) inter-team coordination, and (6) delivery strategy. Each factor represents a category that has a set of practices, which facilitate the alignment of autonomous squads.

The alignment of squads works as enabler for squad autonomy since all squads share the same project. Also, the alignment of autonomous squads is perceived as an umbrella that has a set of practices that enables squad autonomy and mitigates risks of divergence from shared development objectives, while progressing toward achieving squads missions. Fig. 5.1 illustrates the alignment notion among autonomous squads. The findings, which are described in detail in Chapter 5, show a tension between squads autonomy and alignment. Too much alignment might hinder the autonomy of the squads, but at the same time without alignment, the squads are autonomous but are ineffective.

7.4 Challenges to the Heterogeneous Tailoring Approach

The analysis has identified two main challenges to the Heterogeneous Tailoring approach. Firstly, it is difficult to measure the overall code quality, performance, and productivity across squads. Secondly, there is a need for new practices to align and govern enterprise architectural decisions across autonomous squads.

The first challenge is the difficulty of measuring the overall code quality, performance, and productivity for different squads. This challenge emerged because of employing multiple autonomous squads that have different selected and tailored agile development

methods. In this heterogeneous environment, squads tend to tailor agile practices based on their missions. While some squads missions value innovation more than plan fulfilment, others value plan fulfilment more than innovation. A practitioner says: *"Employing a wide range of agile methods (such as Lean Startup, Scrum, Scrumban, Kanban, etc.) and scaling them based the missions of the squads makes it difficult for the organisation to measure the overall velocity of the squads"*–P1, Agile Coach and Architect.

The second challenge is aligning and governing architectural based decisions across autonomous squads. This challenge was a result of having a centralised architectural based decision-making by having an architect. An architect role was employed because of the complexity of this case study project. Practitioners say: *"Deciding about which solution to adopt is time-consuming for the developers despite the employment of Chapters"*–P1, Agile Coach and Architect. Also, *"we discuss newly requested features with the architect or those developers who have experience in specific parts of the project to find the best solution"*–P10, Product Owner. Moreover, *"I get in touch with our architect whenever is need to discuss some user stories and to decide on which architectural change should be employed"*–P9, Senior Developer. Hence, a centralised architectural based decision-making was employed despite the utilisation of Chapters communities.

The findings show that Chapter leaders do not have ownership of the architecture. Most of the stakeholders were referring to the architect directly. In addition, lacking a process for governing architectural based decisions resulted in obstacles to making a generic architectural decision and aligning them across the squads. A practitioner says: *"We lack a process that manages and aligns architectural decisions among the squads while sharing the same product"*–P3, Senior Developer. This lack of process, in turn, impacts the quality being produced while working towards achieving common technical or business roadmap.

7.5 Summary

This chapter has addressed the research question (RQ2): *What is the approach taken to agile tailoring, when using the Spotify model?* To answer this research question, A longitudinal embedded case study was conducted. The data were collected through observing 225 ceremonies over 21 months, conducting 14 semi-structured open-ended interviews, and accessing different sorts of artefacts. The collected data was analysed using an approach informed by the Grounded Theory method.

The analysis revealed a novel approach to agile tailoring, called Heterogeneous Tailoring, by looking at agile tailoring through the lens of Spotify Tailoring. Three key features characterise this approach. Firstly, each autonomous squad is empowered to select and tailor its development method. Secondly, each squad is aligned with other squads and to common product development goals and objectives. Thirdly, product steering committee draws the strategy of squads' missions and aligns the product backlog among autonomous squads.

This chapter has identified some benefits and challenges for the Heterogeneous Tailoring approach. The identified benefits include (1) mitigating challenges of utilising agile practices across squads that do not fit the needs of each squad, (2) improved creativity for some squads and productivity for others, (3) and mitigated risks of divergence from shared development objectives by employing alignment practices. Also, this chapter identified two main challenges to the Heterogeneous Tailoring approach. Firstly, it is difficult to measure the overall code quality, performance and productivity across squads. Secondly, aligning and governing enterprise architectural decisions across autonomous squads requires new practices.

Chapter 8

Spotify Tailoring for Architectural Governance

8.1 Introduction

One of the identified challenges to the Heterogeneous Tailoring approach and the Spotify model is aligning and governing architectural decisions. This chapter addresses the research question (RQ3): *What new architectural governance practices can be introduced for loosely coupled yet cooperating squads?* To answer this research question, an intervention embedded case-study was conducted, as described in Sect. 3.5. In this intervention, I developed a novel approach to architectural governance and evaluated it in FinTechOrg. This approach incorporates a structural change and an architecture change management process.

The findings presented in this chapter, describe the characteristics of the introduced approach to architectural governance, its benefits, and challenges. Also, this chapter identified the impact of the evaluated architectural governance approach on the Heterogeneous Tailoring approach. This chapter describes how the Heterogeneous Tailoring approach was adapted to accommodate the introduced approach to architectural governance.

This chapter is an expansion of two peer-reviewed research papers [96, 97].

8.2 An Approach to Architectural Governance

The introduced approach to architectural governance is driven by the context of the Spotify Engineering Culture in a multinational FinTech organisation with a large-scale project. The developed approach to architectural governance was evaluated in FinTechOrg. During the intervention, direct observation of agile practices was conducted for 3 months. Also, 8 semi-structured open-ended interviews were conducted to understand the reasons behind the occurred results and to collect practitioner perceptions.

This section presents the findings by describing the characteristics of the architectural governance approach. This approach incorporates a structural change and an architecture change management process.

8.2.1 Organisational Structural Change

This intervention introduced a change to the organisational structure. This change aims to facilitate the alignment architectural decisions among squads and ultimately to strengthen the autonomy of squads. The structural change is depicted in (1) empowering Chapter Leaders and experienced developers with the role of Architecture Owners, (2) changing the responsibilities of the architect to be of Enterprise Architectural focus, and (3) locating all Architecture Owners in a virtual squad that is led by an Enterprise Architect. Fig. 8.1 illustrates the employed structural change.

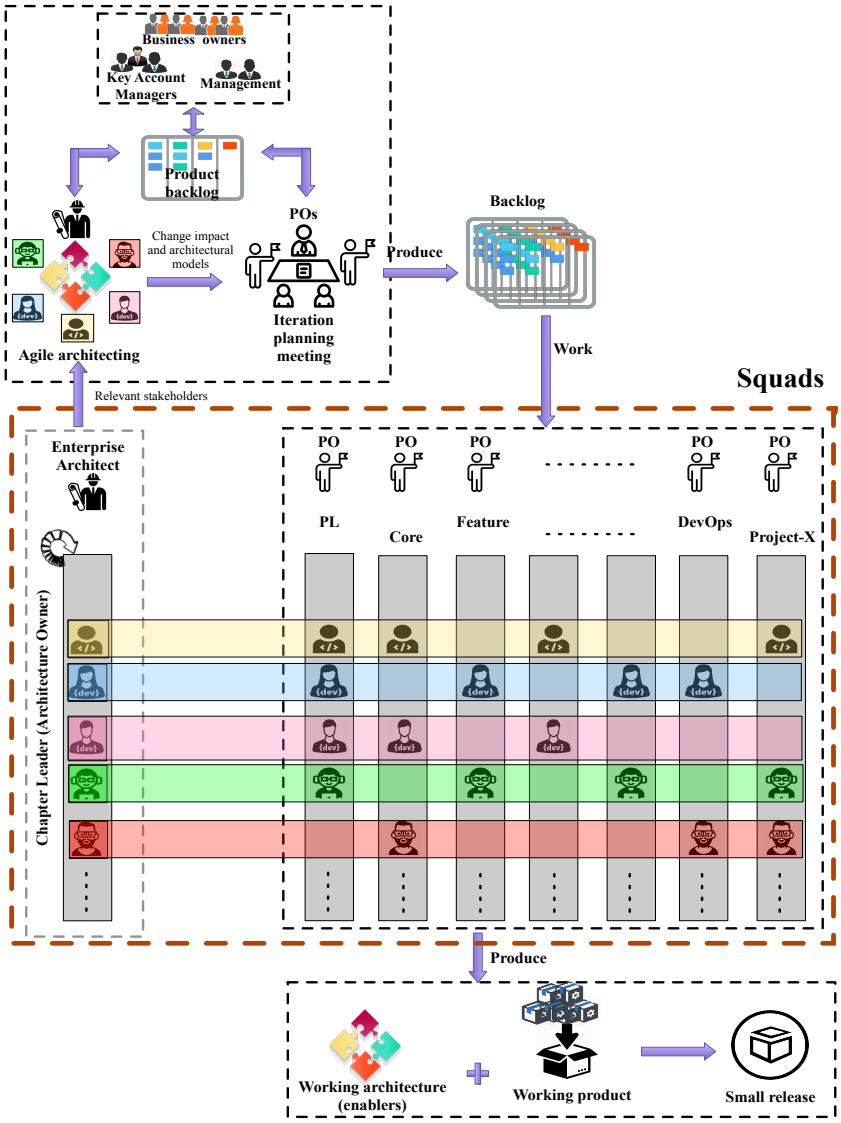


Figure 8.1: Structural change for architectural governance, which is adapted from [96]

Before conducting this embedded case-study intervention, the organisation was utilising an architect role because of the complexity of the FinTech project. The architect was carrying out both of the Architecture Owners and the Enterprise Architect roles. A practitioner says: *"I was the main reference for all squads when it comes to any architectural change in our project... I was taking into consideration the complete value chain and the full lifecycle of the provided software service"*–P19, Enterprise Architect. However, the organisation had challenges in aligning and governing architectural decisions across

autonomous squads. *"The size of the development team is now much larger than what it was 3 years ago... I am overloaded with many responsibilities and tasks which in turn causes a delay, sometimes, in taking decisions or minimises spreading such decisions among other squads in a proper way"*–P19, Enterprise Architect. After conducting this intervention embedded case-study, the role and responsibilities of both the architect and the Chapter Leaders were tailored.

Architecture Owner role and responsibilities

The role of Architecture Owner is assigned to Chapter Leaders and experienced developers. Since Chapters are created based on competency areas and Squads are aligned on the product-level, the architecture owners should be aligned accordingly. A practitioner says: *"Breaking down the role of the architect into Architecture Owners roles and distributing these roles among Chapter Leaders, based on their skills, move architecture decision-making into the development level and align architectural based decisions"*–P20, Agile Coach. Hence, Chapter Leaders, who carry out the role of Architecture Owners, spend most of their time working as developers and spend the rest of their time performing architecture activities besides leading a specific Chapter. However, carrying out the Architecture Owner responsibilities can increase the overhead on Chapter Leaders. A practitioner says: *"Giving me the role of Architecture Owner facilitates taking architectural decisions within my Chapter...Though, this role increases the overhead on me since I work also as a developer"*–P15, Senior Developer and Chapter Leader.

The Architecture Owners' awareness of the business and technical roadmaps is essential for aligning architectural decisions within Chapter based communities and hence across autonomous squads. Practitioners say: *"Chapter Leaders should be aware of our business and technical challenges to work on them along the journey"*–P20, Agile Coach. Also, *"discussing architectural decisions within my Chapter facilitates building an agreement among the squads about how to do stuff and when to do it"*–P17, Senior Developer and Chapter Leader.

Architecture Owners are responsible for bringing valuable architectural knowledge to squads. A practitioner says: *"Since our Chapter Leaders have technical and domain knowledge, they can help their Chapters in taking architectural decisions within their*

expertise"–P18, Product Owner. Architecture Owners create technical guidelines, such as coding and security guidelines, and they mentor and coach the members of their Chapters in architecture and design skills. A practitioner says: *"Spreading technical and architectural based knowledge within my Chapter is one of my responsibilities... I create some coding guidelines, review code, and coach my Chapter whenever needed"*–P15, Senior Developer and Chapter Leader.

Architecture owners are responsible for resolving conflicted architectural decisions and mitigating technical risks across squads. Developers, which are distributed among multiple autonomous squads, might encounter conflicted architectural choices. A practitioner says: *"Many developers are smart and strong-willed where they do not always come to an agreement... Someone should lead and facilitate the evolution of the architecture"*–P15, Senior Developer and Chapter Leader.

Architecture Owners are responsible for exploring challenging architectural tasks. Thus, Architecture Owners might either explore such complex architectural work on their own or could ask a Chapter member, who has encountered this challenge, to explore it. Then, both sides can discuss the details. Practitioners say: *"I use an architectural spike to write just enough code to explore the benefits of a specific technology or technique that other members of my Chapter are unfamiliar with*–P17, Senior Developer and Chapter Leader. Also, *"sometimes, I do ask other members to explorations on their own... Yet, we do discuss the results within our chapter before taking a final decision"*–P15, Senior Developer and Chapter Leader.

Architecture Owners should collaborate closely with the Enterprise Architect and other Architecture Owners within the architecture squads. This collaboration facilitates getting the best out of the Architecture Squad and creates better alignment across the organisation. A practitioner says: *"The main reason behind creating a virtual Architectural squad, which consists of Chapters Leaders who have the Architecture Owner roles, is to have proper technical and architectural based alignment through the organisation... Meeting whenever needed is important to resolve encountered technical or architectural issues"*–P20, Agile Coach.

Enterprise Architect role and responsibilities

The role of Enterprise Architect is assigned to the Architect. Since the architect is overloaded with many responsibilities, the architect's responsibilities were changed to be of enterprise nature. Practitioners say: *"The architect has great knowledge about the technical and the business roadmaps of our organisation... He should continue focusing on the Enterprise architectural tasks"*–P20, Agile Coach. Also, *"our Architect has a solid understanding of various approaches to software development"*–P16, Senior Developer. Leading the Architecture squad, by the Enterprise Architect, demands providing Architecture Owners with the required support and strong commitment. A practitioner says: *"It is crucial to have the required help and support from our the Enterprise Architect in taking enterprise architectural decisions such as integrating two intercorrelated components or even resolving conflicted architectural decisions between the squads"*–P17, Senior Developer and Chapter Leader.

The Enterprise Architect works with solution management teams (i.e., Product Owners and Key Account Managers) and close to the Architecture Owners. This close collaboration was observed to be vital for aligning architectural decisions over the roadmap and solution intent. A practitioner says: *"The Enterprise Architect spends much time collaborating with senior stakeholders across the organisation to create proper technical and architectural alignment across the squads"*–P18, Product Owner. This close collaboration facilitates applying proper enterprise architectural decisions at the right time according to the business values.

The Enterprise Architect focuses on creating architectural and technical alignment for the full software project. In contrast, the Architecture Owners are responsible for specific components within the project and concerned about specific technical competency areas. Practitioners say: *"I started focusing on architectural based tasks that are related to the full solution to create technical and architectural alignment"*–P19, Enterprise Architect. Also, *"Our Architect should be concerned only with Enterprise based architectural decisions instead of wasting his time on small issues that can be tackled within our Chapters"*–P15, Senior Developer and Chapter Leader.

Enterprise Architect is responsible for promoting enterprise architectural practices and driving architectural initiatives. The Enterprise Architect facilitates making enterprise

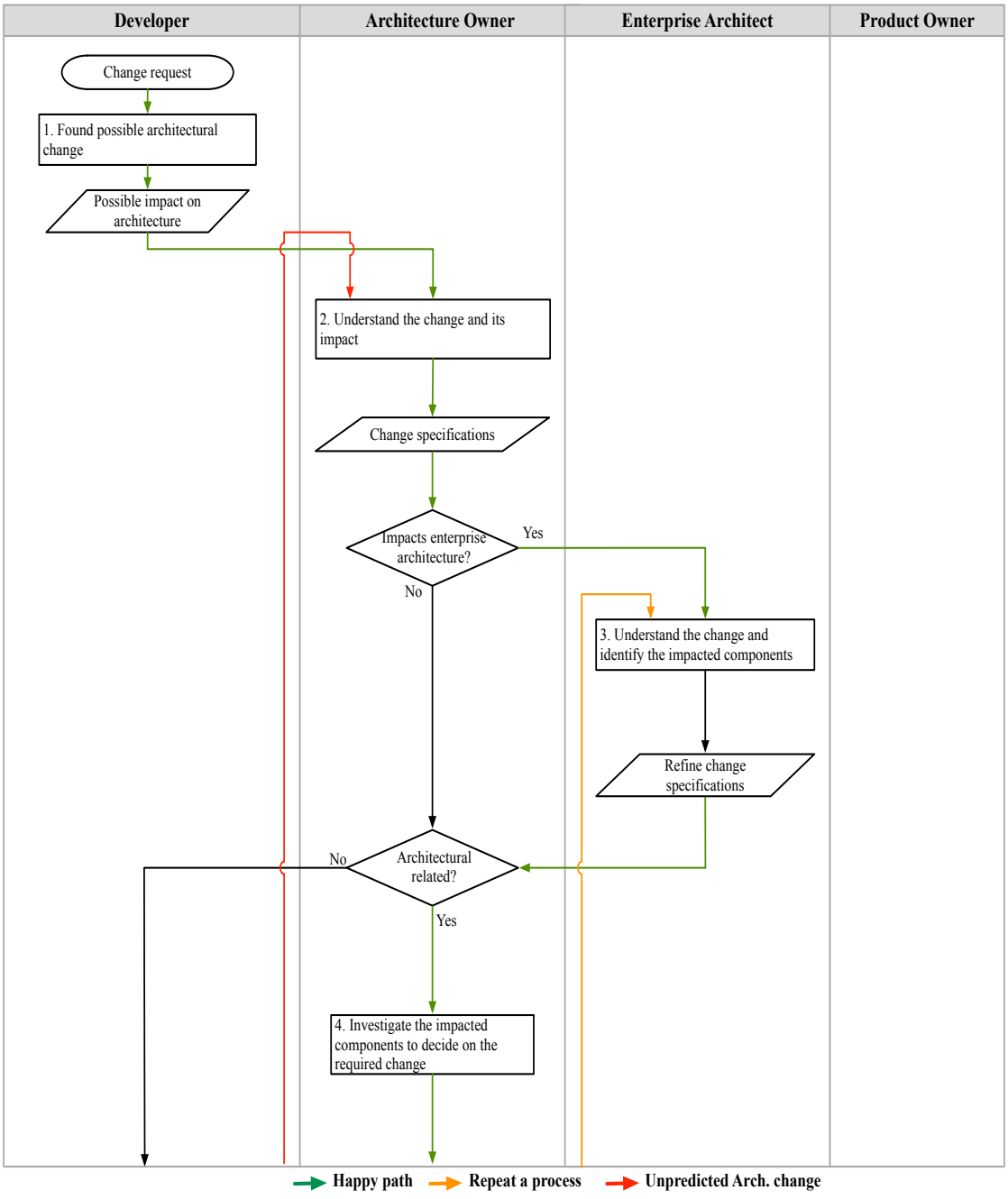
architectural decisions and aligning them across autonomous squads, instead of forcing architecture creation. The Enterprise Architect advises reusing introduced ideas, components, and aligning proven patterns across squads while collaboratively working with squads to develop and evolve the architecture. Practitioners say: *"The Enterprise Architect creates architectural initiatives, promotes architectural practices among the squads, and creates technical and architectural alignment on the enterprise level"*–P20, Agile Coach. Also, *"our Enterprise Architect does not force us adopting a specific architectural decision... Instead, he drives architectural initiatives and facilitates them among us (i.e., Architecture Owners)"*–P15, Senior Developer and Chapter Leader.

8.2.2 Architecture Change Management Process

An architecture change management process was developed before conducting the embedded case-study intervention. This change management process was adapted throughout the intervention case-study. The main objective of this change management process is to guide whoever involved in architectural changes in governing and aligning architectural based decisions. Fig. 8.2 presents the flow of these activities, which are described as follows:

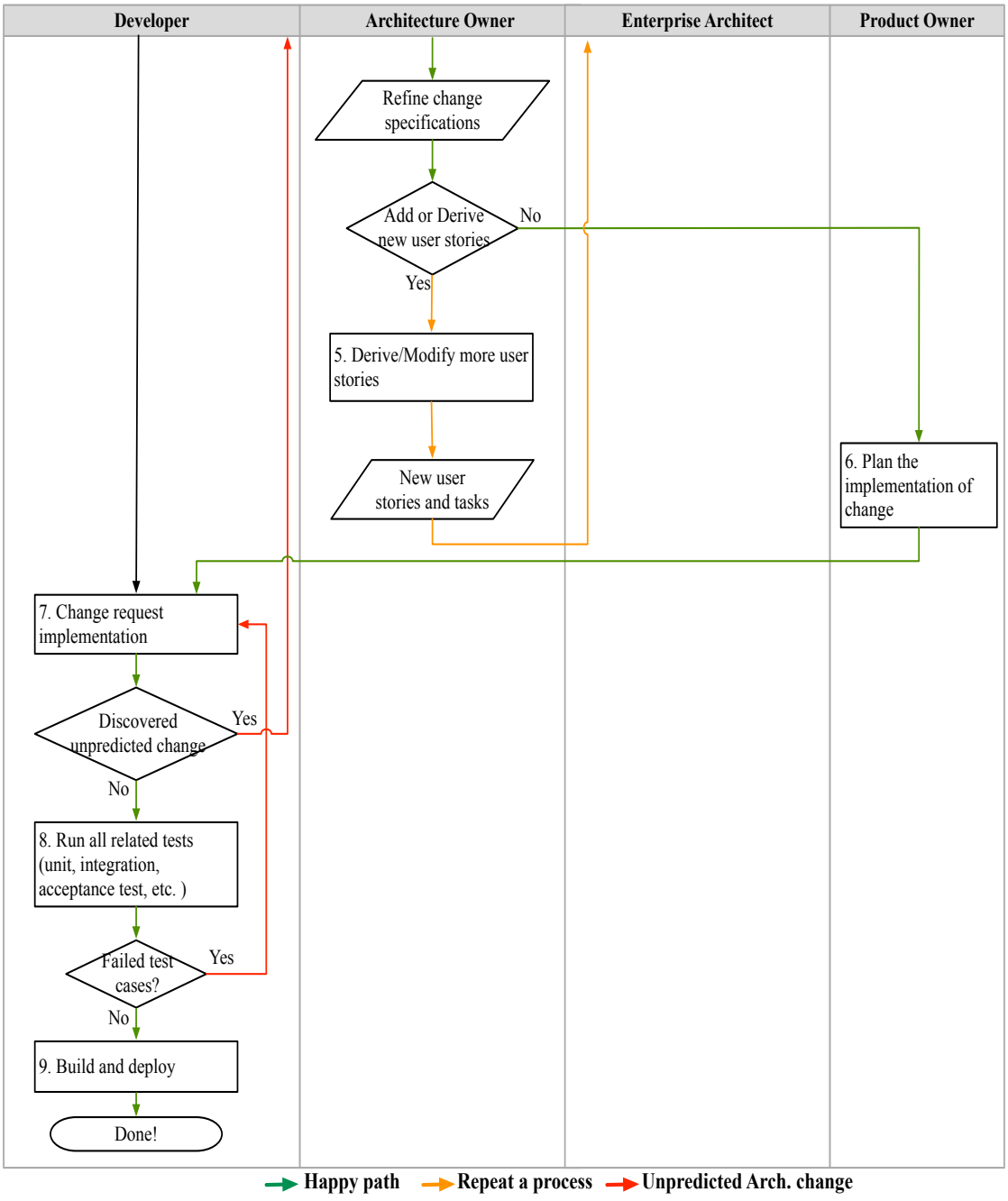
Activity 1: Find possible architectural change: When a developer finds a possible architectural change, the developer should determine the impact of the change request. Identifying the impact on architecture facilitates determining and discussing the change and its impact. In order to initiate the analysis process, the developer involved should create a Kanban card that describe the change request in more technical specifications and visualise it as a WIP in the analysis phase.

Activity 2: Understand the change and its impact on the project and its architecture: The architecture owner and the involved developers should understand the nature of the change and determine its potential impact on the software project and its architecture. The Architecture Owner updates the Kanban card with more accurate technical specifications. The outcome of this activity should provide plausible data about the impacted parts of the project and its architecture.



(a)

Figure 8.2: Change management process for architectural governance – part 1. Adapted from [96, 97]



(b)

Figure 8.2: Change management process for architectural governance – part 2. Adapted from [96, 97]

If the suggested changes can impact the architecture, one of two possible actions should be performed:

1. If the work requires an enterprise architectural change, the architecture owner should discuss the necessary change with the enterprise architect and if needed with the architecture squad– should follow Activity 3. Thus, the card is moved in Kanban board as WIP in the enterprise analysis phase.
2. If the work does not require enterprise architectural change, the architecture owner and the involved developers should determine whether the requested change could affect the architecture or not.
   (a) If the requested change could affect the architecture, the architecture owner and the involved developers should investigate the impacted components and decide on the necessary change, follow Activity 4.
   (b) Otherwise, the task can be forwarded to the responsible squad for implementation – follow Activity 7).

Activity 3: Understand the change and its impact: In this activity, all involved stakeholders – including the developer, Architecture Owner, and Enterprise Architect – should discuss the received data about the impacted parts of the project and its architecture. If the architectural change request impacts other parts of the project, those Architecture Owners who work on the individual impacted part of the project are invited to this session. In this activity, the architecture owner presents the provided data in the Kanban card, which is the outcome of Activity 2. Then, an impact analysis is initiated.

Since it is unlikely that the outcome of this process can get precise impact analysis merely by looking at the architecture design, an iteration-based impact analysis process can be conducted. Hence, further investigation can be initiated later in this change management process to gain sufficient confidence in the perceived impact of the architectural change request.

This activity provides an in-depth data about the nature of requested change and its impact on the architecture. The outcome includes the identification of the impacted components

and the specifications of newly introduced scenarios and requirements. Afterwards, the stakeholders might decide whether to proceed to a low-level investigation (following Activity 4) or to proceed in the change request implementation if an architectural change is rejected and a new solution is introduced to replace the architectural change request (following Activity 7).

Activity 4: Investigate the impacted components to decide the necessary change: When the architecture based stakeholders (from Activity 3 or Activity 2) decide that an architecture change is required, this activity is followed. During this activity, a meeting is carried out in which architecture owner, together with the involved developers, investigate the specifications of the architectural change request deeply. In this meeting, they break down the specifications and requirements into scenarios and document the details into tasks that can be planed. This activity might derive the possibility of identifying new impacted components. Hence, the architecture owner might create new user stories for unpredicted changes.

The architecture owner and the involved developers determine whether they can identify newly impacted components. If it were possible, they follow Activity 5. Otherwise, the Kanban card, which includes the user story and its tasks, should be moved into To-Do state and forwarded to POs for planning, follow Activity 5.

Activity 5: Derive/modify more user stories: When the architecture owner and the involved developers identify newly impacted components, the architecture owner will create new user story about unpredicted changes. Then the architecture owner moves it to WIP in the enterprise analysis phase, which is followed by Activity 3. Thus, a new iteration of the analysis is initiated. In this case, an informal meeting is conducted where such changes are discussed briefly.

Activity 6: Plan the implementation of change: If the change request receives approval from the architecture squad, and the involved architecture owner and developers, the Kanban card should be available for the planning and development. Since the story card is available for planning, POs arrange between themselves to plan the implementation of card tasks. Then, POs can forward the user story and its tasks to the relevant squads for implementation, follow Activity 7.

Activity 7: Change request implementation: In this activity, the implementation of the change is carried out. The squads utilise a hybrid process of Behaviour Driven Development (BDD) and Test Last Development (TLD). Since the user stories describe the behaviour of the introduced scenarios and requirements, developers are expected to implement the described behaviour in their TLD.

After implementing the required code to satisfy the user stories and the described scenarios, other existing test cases might get impacted by newly developed code. For those test cases that are subject to modification, developers should find the related test cases and modify them accordingly. Two cases could encounter the development team while implementing the change request:

1. If the solution is to modify or add new requirements that would require an architectural change, developers should inform the PO and Architecture Owner about the suggested changes to the requirements. In this case, the suggested changes by developers should be declared as unexpected changes. Afterwards, architecture owner and developers should arrange an on-the-fly (informal) meeting to investigate the discovered unexpected changes, follow Activity 2. The meeting should reveal (1) whether or not the suggested changes might introduce unreasonable architectural change and (2) the size of work required to cope with the suggested changes.
2. Otherwise, follow Activity 8.

Activity 8: Run all related tests: In this activity, developers should utilise the continuous integration as a first step to avoid delays caused by integration problems. Subsequently, a continuous testing process should be initiated to obtain immediate feedback on the possibility of violating architectural countermeasures to prevent unreasonable risks associated with a software release. The scope of testing should be extended from a bottom-up assessment (from test cases to behaviour requirements) to validate architectural goals as well as software project behaviour. In case of any violation of requirements after running the continuous testing, developers should check the implementation and failed test cases, follow Activity 7. Otherwise, follow Activity 9.

Activity 9: Build and deploy: This activity should be followed once the continuous testing is completed successfully. A new release can be planned for deployment on production.

8.3 Benefits and Challenges of the Architectural Governance Approach

This section identifies the benefits and challenges of the evaluated approach to architectural alignment and governance.

8.3.1 Benefits

The evaluated approach to architectural governance has transformed architectural decision-making from centralised into decentralised decision-making. This transformation is mainly perceived in devolving architectural decision-making to the operational level through the organisational structural change. A practitioner says: *"I do not need to wait for the architect anymore... Instead, I can get in touch directly with our Architecture Owner"*–P16, Senior Developer. However, enterprise architectural decisions might be discussed within the architecture squad. A practitioner says: *"Taking decisions about how to integrate different components, or APIs might require a deep investigation by multiple Architecture Owners and the Enterprise Architect"*–P15, Senior Developer and Chapter Leader.

Decentralising architectural decisions provided the Enterprise Architect with the opportunity to focus on creating enterprise architectural alignment for the full project. A practitioner says: *"I focus now on aligning the enterprise architectural direction with the solution intent rather than being concerned for specific components"*–P19, Enterprise Architect. This alignment involves making sure that project vision and roadmap are carried out across the architectural based working items. *"The Enterprise Architect ensures that all Architecture Owners support the desired architectural capabilities and directions of the overall solution"*–P15, Senior Developer and Chapter Leader. Thus,

the Architecture Squad utilises the evaluated change management process to have close collaboration and align architectural decisions. A practitioner says: *"The introduced change management process facilitates the alignment of architectural decisions across the whole organisation"*–P17, Senior Developer and Chapter Leader

This approach has facilitated resolving conflicted architectural decisions and mitigating key technical risks among autonomous squads. A practitioner says: *"Many complex technical and architectural aspects are left to evolve through iterative and incremental development and learning"*–P5, Senior Developer and Chapter Leader. Thus, complex technical and architectural decisions are finalised later in the development as depicted in the evaluated change management process. Also, squads are empowered to make local architectural decisions on their own without waiting for the architect. A practitioner says: *"We are encouraged to make architectural decisions on our own with support of our Architecture Owner without waiting for the architect where the technical details are left to evolve"*–P22, Senior Developer. Furthermore, Architecture Owners avoid dictating specific architectural directions in favour of a collaborative team-based approach. A practitioner says: *"I try to encourage our the members of my Chapter to collaborate and discuss architectural based decisions to make the right architectural decisions"*–P5, Senior Developer and Chapter Leader. Moreover, Architecture Owners work closely with the Enterprise Architect to handle enterprise architectural decisions. The Enterprise Architect says: *"Our Architecture Owners get back to me when they encounter a situation that requires making an enterprise based architectural decision... Sometimes we discuss such enterprise architectural changes with other Architecture Owners if needed"*–P2, Enterprise Architect.

The architectural governance approach has facilitated sharing architectural knowledge among autonomous squads. The Enterprise Architect works on transitioning architectural skills and training Architecture Owners. A practitioner says: *"Our Enterprise Architect started arranging and conducting workshops to train and coach our squads in architectural related aspects"*–P15, Senior Developer and Chapter Leader. Also, Architecture Owners focus on creating architectural alignment and disseminating architecture knowledge. Practitioners say: *"we provide technical guidance around coding, security, architectural based aspects, monitoring the work, and so on"*–P17, Senior Developer and

Chapter Leader. Also, *"Our Architecture Owner, arranges sometimes formal classes and other times brown-bag lunch sessions... In such sessions, we discuss planned subjects of interests which are related to our Chapter"*–P16, Senior Developer.

The evaluated approach to architectural governance has improved software quality and mitigated obstacles to aligning architectural decisions across autonomous squads. FinTechOrg works on balancing the effort for architecting, by utilising the introduced change management process while having a decentralised architectural based decision-making. The architecting effort is distributed throughout the organisation (vertically and horizontally) among the Enterprise Architect, Architecture Owners, and Chapters. This balance of agile architecting effort, in turn, facilitates the creation of generic software features, minimises wasted effort in architectural refactoring, and governs the architecture while strengthening squads autonomy. Practitioners say: *'Conducting proper architectural analysis within our Chapter and then evaluating and discussing the results, if needed, with the Enterprise Architect improves the quality of our produced work"*–P15, Senior Developer. However, *"It was time-consuming to take a good architectural decision that can be maintained easily in future without making much refactoring... overlooking some aspects that can be considered at the time being can cause a lot of waste because of the needs for refactoring"*–P21, Senior Developer.

8.3.2 Challenges

The practitioners, in FinTechOrg, reported two main challenges of the evaluated approach to architectural governance. Firstly, the planning activity is impacted negatively by prioritising user stories without considering the architectural aspects. The change management process does not support screening the user stories by the Architecture Squad before or during the planning activity. A practitioner says: *"We do not go through the user stories, in our Architecture Squad, before planning... Yet, sometimes we discuss them informally upon POs request"*–P17, Senior Developer and Chapter Leader. However, the change management process handles such a situation by moving from activity 6 to activity 1. A practitioner says: *"The architecture squad does not join our planning session... We expect that our the squad members should initiate the process of investigation and provide good input to either of the Architecture Owner or the Enterprise Architect"*–P18, Product Owner.

Secondly, handling architectural spikes might require making provisional architectural decisions or even multiple suggested decisions to explore. FinTechOrg considers an architectural spike as an investment to explore what should be built and how to build it. Practitioners say: *"We allocate some resources for complicated work items, ahead of the targeted delivery deadline, to find out what needs to be done... Such investments are considered a necessity to solve architectural issues, which work as an enabler for the next Sprint"*–P18, Product Owner. Also, *"sometimes when we discuss a user story with the enterprise architect and the architecture owner, the outcome can be ambiguous as there is no concrete decision that can be taken... We have to explore multiple solutions"*–P22, Senior Developer. Hence, Architecture Owners or senior developers can explore the architectural change by writing enough code to investigate proposed solutions. This exploration process can be expensive for complicated architectural spikes. A practitioner says: *"Architecture owners might pair with another developer to explore complicated architectural spikes"*–P19, Enterprise Architect. Hence, practitioners might need to employ an iterative and incremental development to evolve the architecture by utilising the introduced change management process. This process can be time-consuming and yet powerful technique for risk-reduction.

8.4 Adapting the Heterogeneous Tailoring Approach for Architectural Governance

This section presents how the Heterogeneous Tailoring approach was adapted to accommodate the evaluated approach to architectural governance. This adaptation has impacted the key features of the Heterogeneous Tailoring approach. Fig. 8.3 illustrates the impact of the architectural governance approach on the Heterogeneous Tailoring approach.

Product development, in the Heterogeneous Tailoring approach, is affected by the introduced approach to architectural governance. This impact is perceived in the outcome of the employed change management process, which produces new architectural based user stories (i.e., enablers) in the backlog. These enablers should be prioritised and planned along to the rest of the backlog items. A practitioner says: *"This change management*

process results in sometimes new architectural based tickets that need prioritisation and planning"–P18, Product Owner.

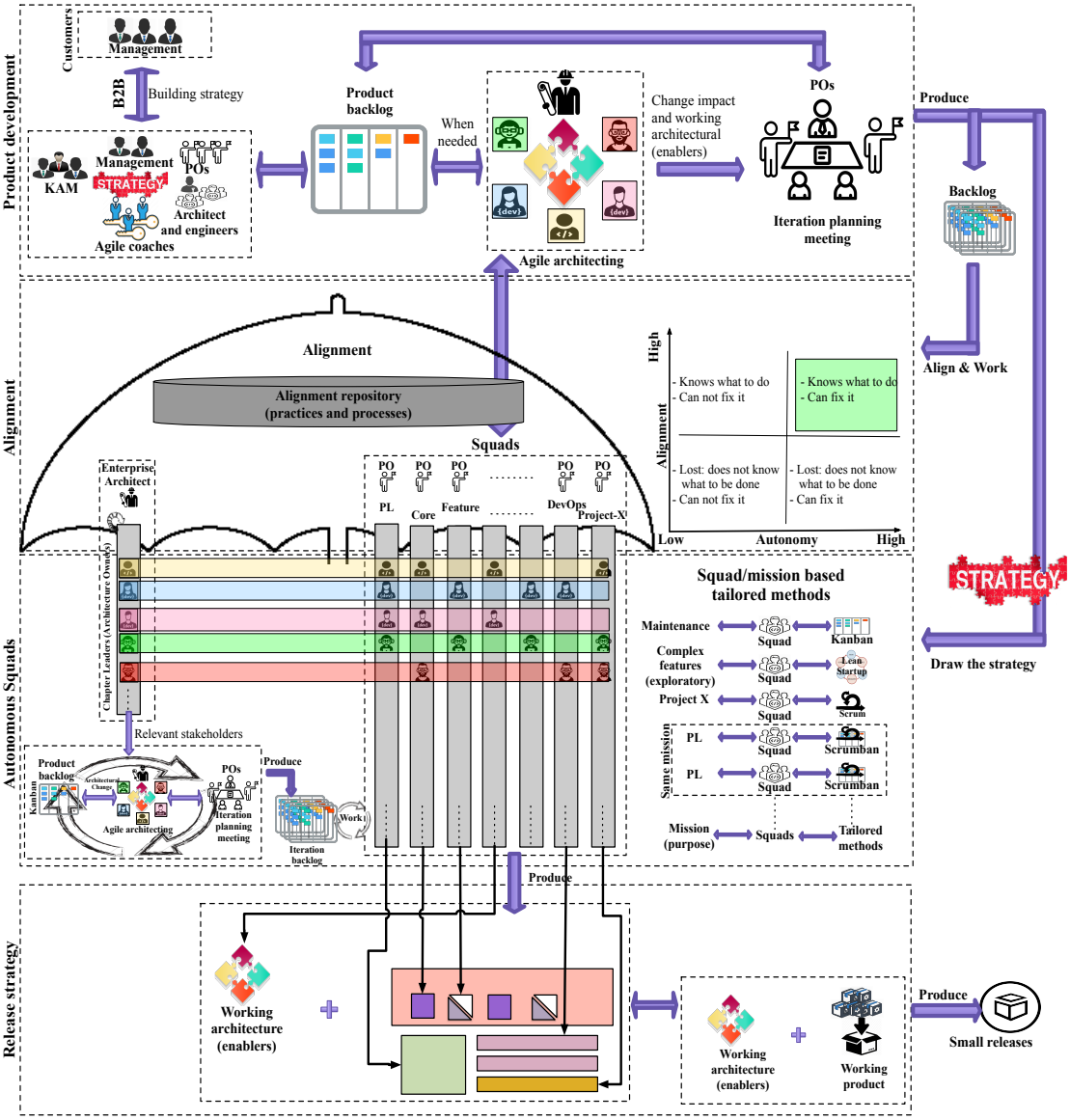


Figure 8.3: Adapting the Heterogeneous Tailoring approach for architecture governance. Adapted from [96]

The alignment of autonomous squads was improved by aligning architectural decisions. This alignment is depicted in both of the structural change and the change manage-

ment process. The structural change has devolved architectural decision-making to the operational-level through Chapters communities. A practitioner says: *"Discussing architectural based aspects with our Architecture Owner speeds up the process and puts all members of our Chapter on the same page"*–P21, Senior Developer. The introduced change management process has aligned the agile architecture among the involved stakeholders (developers, Architecture Owners, Enterprise Architect, and Product Owners) to enable squads autonomy. Practitioners say: *"Following this standardised process across the squads helps all parties to organise the architectural work instead of being dependant on some members from squads or even relying only on the architect"*–P20, Agile Coach. Also, *"The nature of the backlog items requires responding quickly to our customers... The process is now more disciplined and organised, which consequently increased the speed of taking architectural decisions"*–P22, Senior Developer.

The autonomy of the squads was improved, in FinTechOrg, by utilising this architectural governance approach. The introduced approach has strengthened squads autonomy by decentralising architectural based decision-making, which is depicted in the introduced structural change. Also, an architecture change management process was introduced to balance the agile and architecture based disciplinaries. Balancing these disciplinaries, by employing a structural change and an architecture change management process, has created a holistic approach that enables squads autonomy. A practitioner says: *"The employed rules in our architecting process balance the agile and architecture activities... These rules along to the structural changes have mitigated the dependencies and in turn improved the autonomy of the squads"*–P20, Agile Coach.

Governing architectural decisions impact the employed release strategy in the Heterogeneous Tailoring approach. This impact is perceived in continuous delivery of both architecture enablers and working product. A practitioner says: *"We release developed architectural based tickets whenever they are finished... Releasing such tasks enables the upcoming sprints"*–P21, Senior Developer.

8.5 Summary

This chapter has addressed the research question (RQ3): *What new architectural governance practices can be introduced for loosely coupled yet cooperating squads?* An intervention embedded case study was conducted to develop and evaluate a novel approach to architectural governance. The data were collected through observing 32 ceremonies over 3 months and by conducting 8 open-ended semi-structured interviews. The collected data was analysed using an approach informed by the Grounded Theory.

This Chapter has presented the characteristics of the evaluated approach to architectural governance in its final state after carrying out the embedded case study intervention. This architectural governance approach incorporates an organisational structural change and an architecture change management process. The structural change is depicted in (1) empowering Chapter Leaders and developers with the role of Architecture Owners, (2) changing the responsibilities of the architect to be of Enterprise Architectural focus, and (3) locating all Architecture Owners in a virtual squad that is led by an Enterprise Architect. The introduced change management process comprises a set of activities that facilitates architectural governance and alignment among autonomous squads.

The practitioners, in FinTechOrg, have reported benefits of this introduced approach to architectural governance. These benefits include: (1) devolving architectural decision-making to the operational-level, which in turn transformed architectural decision-making into decentralised based decision-making, (2) creating technical and enterprise architectural alignment for the full software project, (3) resolving conflicted architectural decisions and mitigating key technical risks across autonomous squads, (4) sharing architectural knowledge among the squads, (5) minimising wasted effort in architectural refactoring, (6) balancing the effort for architectural quality facilitates the creation of generic software features, and (7) improving software quality and mitigating obstacles to aligning architectural decisions across autonomous squads. Consequently, the alignment and governance of architectural decisions have improved the autonomy of squads.

However, the practitioners reported two challenges of the evaluated approach to architectural governance. Firstly, the planning activity is impacted negatively by prioritising user

stories without considering the architectural aspects. Secondly, handling architectural spikes might require making provisional architectural decisions or even suggest multiple architectural decisions to be explored.

This Chapter, also, presented how the Heterogeneous Tailoring approach was adapted to accommodate the evaluated approach to architectural governance. This adaptation has impacted the key features of the Heterogeneous Tailoring approach. Establishing autonomous squads was impacted by the introduced structural change. The alignment of autonomous squads was impacted by governing agile architecture. Product development was impacted by the needs for planning architecture based user stories. Also, the adaptation of the Heterogeneous Tailoring approach has identified a new key feature, called the release strategy, which is concerned with the continuous delivery of architecture enablers.

Chapter 9

Discussion

This chapter answers the research questions, analyses and interprets the findings in the light of related literature, and discusses the limitations of this research.

9.1 Research Questions & Answers

There is a conflicting trade-off between alignment and autonomy in the Spotify model. Therefore, this research study has explored: **RQ1:** *How do Agile practitioners, from a FinTech organisation, describe approaches to the balance between squad alignment and autonomy?* Autonomy shifts the authority of decision making to the operational level to improve the motivation and innovation of squads (i.e., teams) and consequently to speed up the process of software delivery [57, 58, 76]. However, squads should be autonomous but don't sub-optimise. Thus, autonomous squads should be aligned with each other and the overall product goals and plans. The alignment of autonomous squads works as enabler for the autonomy of squads [57]. However, there is a conflicting trade-off between autonomy and alignment. Too much alignment might hinder squad autonomy, but at the same time without alignment, the squads can be autonomous yet not effective.

My research has discovered a tension between squads autonomy and alignment. The

alignment of autonomous squads works as enabler for the autonomy of squads. Therefore, Agile practitioners should get the balance right between squads autonomy and alignment. This research has resolved the conflicting trade-offs between autonomy and alignment by identifying influential factors on 3 different aspects of Spotify Tailoring in a FinTech organisation. These aspects are (1) establishing and building autonomous squads, (2) aligning autonomous squads, and (3) performing B2B product development in a FinTech context.

RQ1.1: *How do Agile practitioners, from a FinTech organisation, establish and build squad autonomy?* My research identified factors that facilitate building autonomous squads. These influential factors are (1) tailoring the adaptive structure and creating sufficient communities around it, (2) building squad or mission-based strategy, and (3) tailoring Agile methods on the squad or mission level. Each factor is supported by a set of practices and attributes that can improve squad autonomy. The identified factors and their related practices can aid Agile practitioners in building autonomous squads.

RQ1.2: *How do Agile practitioners, from a FinTech organisation, achieve and sustain the alignment of autonomous squads?* My investigation of Spotify Tailoring, from a squad alignment perspective, has identified new, modified and previously introduced practices and attributes to the Spotify model. Also, my research has identified influential factors on aligning autonomous squads. These influential factors are (1) adaptive structure, (2) collective code ownership, (3) collective decision-making, (4) knowledge sharing, (5) inter-team coordination, (6) mission-based planning, and (7) delivery strategy. Each factor is supported by a set of practices that can strengthen the alignment of autonomous squads. The identified influential factors and their related practices can aid Agile practitioners in aligning autonomous squads.

RQ1.3: *How do Agile practitioners, from a FinTech organisation, tailor the Spotify model for B2B product development?* Two main aspects related to product development were revealed. Firstly, The impact of product development on squads autonomy and alignment was identified. This impact is mainly realised in the identification of squads' missions strategy and the alignment of product backlog items among autonomous squads. Secondly, performing B2B product development, by tailoring the Spotify model, was

identified. My investigation of Spotify Tailoring, from a B2B perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. Also, my investigation has identified influential factors on Spotify tailoring for B2B product development. These influential factors are (1) project visibility, (2) interactions within product development, (3) building relationships, and (4) response time. Each identified factor is supported by a set of practices and attributes that can facilitate B2B FinTech product development. The identified factors and their related practices can aid Agile practitioners in performing B2B product development by tailoring the Spotify model.

This research identified a lack of previous research on the Agile tailoring approach used in the Spotify model. Thus, I have explored: **RQ2:** *What is the approach taken to Agile tailoring, when using the Spotify model?* My analysis in this research revealed a novel approach to Agile tailoring, called Heterogeneous Tailoring. Three key features characterise this approach. Firstly, each autonomous squad is empowered to select and tailor its development method. Secondly, each squad is aligned with other squads and to common product development goals and objectives. Thirdly, product steering committee draws the strategy of squads' missions and aligns the product backlog among autonomous squads.

Two main challenges to the Heterogeneous Tailoring approach were identified by this research. Firstly, it is difficult to measure the overall code quality, performance and productivity for different squads. Secondly, aligning and governing architectural decisions across autonomous squads requires new practices.

Consequently, this study has addressed the second challenge to the Heterogeneous Tailoring approach by answering: **RQ3:** *What new architectural governance practices can be introduced for loosely coupled yet cooperating squads?* An embedded case study intervention was conducted to facilitate architectural governance using the Spotify model. In this intervention, I have developed and evaluated a novel approach to architectural governance. This architectural governance approach incorporates an organisational structural change and an architecture change management process. The structural change (1) empowers Chapter Leaders and experienced developers with the role of Architecture Owners, (2) changes the responsibilities of the Architect to be of Enterprise Architectural

focus, and (3) locates all Architecture Owners in a virtual squad that is led by an Enterprise Architect. The introduced change management process comprises a set of activities that facilitates architectural governance and alignment among the stakeholders. Based on the intervention results, I have adapted the Heterogeneous Tailoring approach to accommodate the evaluated approach to architectural governance in FinTechOrg.

9.2 Discussing the Results

Large-scale agile software development is challenging since several teams need to work closely together to release a single software product while multiple teams need to communicate, collaborate and coordinate [23, 26]. Some identified challenges to large-scale Agile development include maintaining teams autonomy [23], aligning self-organising teams through coordination [26, 79], knowledge sharing [79], and other aspects [23, 26, 79]. The Spotify model is an example of a large-scale Agile engineering culture, which is driven by creating loosely coupled (i.e., autonomous squads), yet tightly aligned squads [57, 58]. The Spotify model employs autonomous squads to have motivated and innovative squads that can deliver quickly [57, 58]. Hence, the authority of decision making is moved to the operational level. The alignment of autonomous squads works as an enabler for the squad autonomy.

However, my research study observed a conflicting trade-off between squads autonomy and alignment, which lacks a previous research and needs exploration. Too much alignment might hinder squad autonomy, but without alignment, the squads are autonomous but are not effective. To this end, a longitudinal embedded case study was conducted, in a multinational FinTech organisation using the Spotify model, to explore how the conflicting trade-offs between alignment and autonomy is resolved.

I have discovered a tension between squad autonomy and alignment, based on my observations and practitioner perceptions. For example, the practitioners reported that collective code ownership requires an alignment over the product-level and demands a sufficient reconciliation process among autonomous squads. Also, the automation of

standardised tasks strengthens collective code ownership and hence strengthens squad autonomy. Another example is the utilisation of sufficient inter-team coordination process to prevent excessive processes that might waste resources and hence impact negatively the autonomy of squads.

9.2.1 Resolving the Conflicting Trade-off Between Squads Autonomy and Alignment

Agile practitioners should get the balance right between squads autonomy and alignment. My rigorous analysis has resolved the conflicting trade-offs between autonomy and alignment by identifying influential factors on 3 different aspects of Spotify Tailoring. These aspects are (1) establishing and building autonomous squads, (2) aligning autonomous squads, and (3) performing B2B product development by tailoring the Spotify model.

Building and maintaining teams autonomy is considered challenging in large-scale agile [23], yet was identified as one of the Agile success factors [20, 26]. Autonomous teams should be composed of cross-functional teams that can cover the full development cycle of a project [55]. Previous research found that autonomous teams should be accountable to committed work [48, 55]. More specifically, squads should be held accountable for achieving their missions since the organisation has put squads' strategy into practice and specified clear expectations from each squad based on its mission [71]. Also, the squads have freedom in determining how to achieve their missions. My research acknowledges the importance of building sufficient team autonomy and maintaining it to utilise effective development. However, the Spotify model does not provide guidelines to how squads should build their autonomy. Nevertheless, the Spotify model utilises a two dimensional structure and creates communities around them [56, 57, 58].

My careful analysis of practitioners' perceptions and my observation identified influential factors on building autonomous squads. These factors are (1) tailoring the adaptive structure and creating sufficient communities around it, (2) building squad or mission-based strategy, and (3) tailoring Agile methods on the squad or mission level. Also, my research found that Tribes communities are inapplicable and Guilds communities and

their meetings pose a waste of resources in large-scale agile (<100 people distributed over 2-9 squads). Guilds communities are replaced by demo sessions and non-routine meetings, as stated by the practitioners. These meetings are announced through Slack or Email, and those who are interested can join the meetings. Each squad should have a precisely defined strategy, which can be the same for multiple squads of the same mission. This strategy facilitates squads' decisions for tailoring Agile practices based on squads' strategy and/or mission.

Aligning autonomous teams is considered challenging in large-scale agile [26, 79]. Agile teams must have common objectives and should be able to organise themselves to overcome encountered challenges [48, 55]. Also, interfacing, coordination, knowledge sharing, and alignment between teams are considered challenging in large-scale Agile development [23, 26, 79]. Mankins and Garton [71] identified balancing squads' autonomy to innovate versus following proven routines to get both outcomes as a challenge when using the Spotify model. Also, it is challenging to balance alignment with control to ensure that coordination and connectivity happen among Spotify squads without relying on controlling managers [71]. Despite Spotify's awareness of the significant role of alignment, it does not provide guidelines for aligning autonomous squads nor factors influential on aligning autonomous [57, 58].

My careful analysis presented Spotify Tailoring, from squads alignment perspective, by identifying new, modified and previously introduced practices to the Spotify model. Also, my careful analysis identified influential factors on aligning autonomous squads. These influential factors are (1) adaptive structure, (2) collective code ownership, (3) collective decision-making, (4) knowledge sharing, (5) inter-team coordination, (6) mission-based planning, and (7) delivery strategy. Each identified factor is supported by a set of practices and attributes that can aid Agile practitioners in aligning autonomous squads.

In relation to my identified influential factors on aligning autonomous squads, previous research found that collective code ownership can encourage all teams to contribute to software development [81]. The autonomy of teams has a direct influence on teams' effectiveness since the authority of decision making is moved to the operational level, which in turn increases the development speed and accuracy of problem-solving [76].

However, I found that utilising collective decision-making on the enterprise level works is needed to align enterprise decision-making. The horizontal alignment through Chapters communities facilitates knowledge sharing and removes the encountered impediments by the squads [57, 58]. As projects grow in size and complexity, the number of inter-team dependencies also tend to increase and hence inter-team coordination became inevitable to resolve dependencies and prioritise work [26, 59, 74].

The Spotify organisation follows the B2C model since it provides music streaming service to end-users and hence there is a provision of guidelines in the grey literature [56, 57, 58, 69]. The Spotify model encourages following Lean Startup for product development since it values innovation [57, 58]. For large-scale FinTech projects with B2B environment, however, there is no guidelines about how to perform product development in a global context. B2B environment is deemed as complicated because of having conflicting agendas, politics and different priorities between two or more organisations [100].

My analysis revealed two aspects related to product development. Firstly, I have identified the impact of product development on squads autonomy and alignment. The autonomy of squads is impacted by the product steering committee, who identify the strategy of each squad and it mission. The alignment of squads is impacted by the alignment of product backlog items among autonomous squads. Secondly, I have identified influential factors on Spotify tailoring for B2B product development: (1) project visibility, (2) interactions within product development, (3) building relationships, and (4) response time. Also, my detailed analysis of Spotify Tailoring, from a B2B perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. Each identified factor is supported by a set of practices and attributes that can aid Agile practitioners in performing B2B product development by tailoring the Spotify model.

9.2.2 Heterogeneous Tailoring Approach

Researchers have long believed that approaches to Agile tailoring can fit into Template-based and Automatic Tailoring [42], Contingency Factors or the Method Engineering approach [17, 24, 35], or Hybrids of Agile and plan-driven approaches [38, 45, 52]. I

found, through literature analysis, that Template-based and Contingency Factors are similar since multiple methods or processes should be available in the organisation, and then a selection is performed based on the features of the project context. Also, Automatic Tailoring is similar to Method Engineering since the creation of new methods is based on existing method fragments or meta-method processes that are applied in specific project context or situation. Therefore, I map these pairs together in Table 9.1, which describes the aims, advantages, and disadvantages of each identified approach.

Other identified approaches for method tailoring fall under either of Contingency Factors or Method Engineering, according to my literature analysis. For example, the proposed frameworks and metamodels for situational method engineering [6, 32, 46] fall under the umbrella of Method Engineering approach. Also, the proposed approach, by Casare et al. [19], for tailoring software development processes according to project needs is based on Method Engineering techniques.

The proposed assessment approaches for adopting Agile methods and practices can fall into either of Contingency Factors or the Method Engineering. For instance, Sidky Agile Measurement Index [3] facilitates the adoption of Agile practices by considering an Agile maturity model. This approach facilitates the creation of new Agile methods based on the targeted maturity model and hence falls under the Method Engineering approach. However, the assessment framework proposed, by Soundararajan et al. [107], falls under the Contingency Factors approach since this framework examines the applicability of various Agile methods for the organisation based on different factors.

My research study has identified a novel approach to Agile tailoring, which does not fit into any previously identified tailoring approach. I have observed, in FinTechOrg, the existence of multiple autonomous squads working on the same project and yet employing different tailored Agile methods. My rigorous analysis of the collected data revealed a novel approach to Agile tailoring, which does not fit into Contingency Factors, Method Engineering, or Hybrid of Agile and plan-driven approach. I call this Agile tailoring approach as Heterogeneous Tailoring.

Table 9.1: Software method tailoring approaches

Approach	Aims	Advantages	Disadvantages
Contingency Factors or Template-based Tailoring	Selects multiple methods to be available in the organisation and then performs the selection based on the features of the project context.	- Overcomes the limitations of Agile methods as there is no sufficient method for all cases. - Follows predefined criteria for method selection, from a portfolio of predefined methods, based on the targeted development context.	- Team members should have a good understanding of a range of methods to adopt this approach. - Team members should be capable of executing a range of method according to the contingencies.
Method Engineering or Automatic Tailoring	Suggests the creation of new methods based on existing method fragments or meta-method processes to be applied in specific contexts.	- Concentrates on activities related to Agile methods and creates in-house methods. - Brings flexibility to the organisation - Responds to challenges of real projects and their context in an efficient way instead of relying on existing methods.	- Method fragments need control to assemble the new method. - Tools are needed for method construction. - Quality evaluation models are required for newly introduced methods. - Knowledge on how to handle specific situations and how to define method fragments are needed.
Hybrid Tailoring	Blends agile methods into traditional plan-driven project development and management.	- Produces high-quality software. - Reduces the risk of upfront ambiguities in terms of the project goals and deliverables. - Speeds up the iterative thinking process and reduce the risk of rework and delay.	- Tailoring the process is difficult and time consuming to change after constructing the hybrid process. - Keep project control from falling into bureaucratic behaviours and contracts.
Heterogeneous Tailoring	Separates the concerns of each team in terms of Agile tailoring. This approach empowers teams to tailor their Agile practice and yet align teams to each other and to common product development goals.	- Improves teams' autonomy by empowering teams to tailor their agile practices. - Improves the creativity or productivity of autonomous teams by the ability to evolve their own Agile practices. - Mitigates the risk of being obliged to adopt shared Agile practices that do not meet the needs of a particular team. - Mitigates the risks of being diverted from shared development objectives.	- It is difficult to measure the overall code quality, performance and productivity across squads.

This Heterogeneous Tailoring approach does not create new Agile methods based on existing method fragments that can be applied to specific contexts, as in the Method Engineering [24, 46]. Also, my approach does not select multiple methods to be available and then perform method selection based on the features of the development context, as in Contingency Factors [24, 35].

Moreover, the Heterogeneous Tailoring approach does not advocate the utilisation of Agile practices across squads that do not fit the needs of each squad. On the contrary, this Heterogeneous Tailoring approach separates the concerns of each squad in Agile tailoring and yet align them to each other and to common product development goals. In this Heterogeneous Tailoring approach, Agile Coaches use the tailoring criteria [17, 54, 114] by considering the different categories, which are summarised in Table 2.1, based on the identified mission for each squad.

9.2.3 An Architectural Governance Approach

Despite the reported benefits of the Heterogeneous Tailoring approach, my research identified a challenge in aligning and governing architectural decisions among autonomous squads. Software architecture is perceived as a key factor to scale up Agile development in a large-scale. The coexistence of software architecture and Agile development was realised by previous research [2, 15, 27, 61]. This coexistence caused the emergence of some Agile architecting practices [118]. However, there is a lack of guidance on how Agile practitioners can use the combination of architecting in the architecture-agility combination. The Spotify model is an example of an Agile model that lacks such guidelines.

To overcome the challenge of architecture governance in the FinTech's Spotify model, an embedded case study intervention was conducted. In this intervention, I have developed and evaluated a novel approach to architectural governance. This architecture governance approach incorporates a structural change and an architecture change management process. The structural change (1) empowers Chapter Leaders and experienced developers with the role of Architecture Owners, (2) changes the responsibilities of the Architect

to be of Enterprise Architectural focus, and (3) locates all Architecture Owners in a virtual squad that is led by an Enterprise Architect. The introduced architecture change management process comprises a set of activities and practices that facilitate architectural governance.

Bellomo et al. [11] identified some Agile architecture patterns and tactics. However, they do not provide a framework or an approach for Agile architectural governance and alignment. In a more abstract level, Nord et al. [82] explored architectural tactics that can improve the alignment of the architecture and the development. Nord et al. proposed an Agile architecture alignment using vertical and horizontal decomposition of the software architecture and matrix augmented-role team structures by conceptualising Scrum as an example. However, the proposed alignment by Nord et al. is not supported by a scientific investigation following a rigorous research process and only conceptualises the proposed alignment using Scrum. My work does so by conducting an empirical investigation of the proposed approach in a specific domain (i.e., FinTech) of large-scale, which tailors the Spotify model.

Yang et al. [118] conducted a systematic mapping study on the combination of Agile development and software architecture and hence identified 11 architecting activities and 41 Agile practices that are used with software architecture. The Spotify model does not provide guidelines for aligning Agile architecting across autonomous teams [98]. My architecture governance approach utilises some Agile activities identified by Yang et al. in the introduced architecture change management process, and introduces new roles within the organisation. The architecting activities used in my architecture change management process are Architectural Analysis and Synthesis (Activity 1 and 2), Architectural Evaluation and Impact Analysis (Activity 3), Architectural Refactoring (Activity 6), Architectural Maintenance and Evolution (moving from Activity 6 back to Activity 1). However, the activities of Architectural Description and Understanding are used to some extent at the enterprise architecture level. Also, Architectural Reuse is practised, based on my observation, within the squads and encouraged by Architecture owners.

Martini and Bosch [73] identified 4 sorts of teams, 3 architect roles, and some architecture practices, which are used to propose and evaluate a framework for Agile architecting

embedded software projects. Their framework employs an Architect within each team, which works closely to a Governance Architect who functions as a coordinator between the Chief Architect and the Agile teams. Also, their framework does not consider having a cross-functional team working on different parts of the software project, which may employ different technologies and different architectures. Therefore, it is unlikely that a single Team Architect (i.e., Architecture Owner) would possess the required architectural knowledge in all technologies used in different parts of the same project and in other industries than embedded software projects. Consequently, Martini and Bosch's framework does not consider aligning the teams horizontally, which is the case in my approach, where it aligns the teams horizontally through Chapters.

Based on the intervention results, I have adapted the Heterogeneous Tailoring approach to accommodate the evaluated approach to architectural governance. This adaptation has impacted the key features of the Heterogeneous Tailoring approach. For example, establishing autonomous squads was impacted by the introduced structural change. The alignment of autonomous squads was impacted by governing Agile architecture. Product development was impacted by the needs for planning architecture based user stories. Also, the adaptation of the Heterogeneous Tailoring approach has identified a new key feature, called the release strategy, which is concerned with the continuous delivery of architecture enablers.

9.3 Limitations

The research limitations are discussed in this section by considering the criteria introduced by Lincoln and Guba [68] for evaluating the trustworthiness of the findings in qualitative research.

Credibility refers to the level of compatibility between the respondents' opinions and the researcher's interpretation and presentation [68]. A longitudinal and intervention embedded case studies were conducted to have in-depth insight into the Spotify model and to overcome one of the identified challenges. Data were collected through semi-structured

open-ended interviews and supplemented by direct observation, documentation, and artefacts. The interview protocol was revised after conducting two pilot interviews. The collected data was analysed using an approach informed by the Grounded Theory (Glaserian approach) [39, 40]. This triangulation in data collection and analysis has provided the researcher with different angles towards the studied phenomenon and thus provided a broader picture. Also, this triangulation has increased the precision of this empirical research and hence improved the construct validity. However, limited description of the explored project was provided because of the confidentiality agreement with FinTechOrg.

Dependability is concerned with the ability to replicate the conducted research [68]. This research was limited to a specific context, as described in Sect. 3.4.1, which is dictated by the research destination of which the researcher works in as senior software engineer. Also, the selection of participant interviewees was limited by their willingness to participate in this research.

Incorrect data is a validity threat since this research study is empirical and data collected from interviews is known to be prone to bias [87]. The presented data by the interviewees to the researcher have 4 types. These types are (1) Baseline data, which represent the best description a participant can provide, (2) Properline data, which represent a description of what participant thinks it is proper to tell the researcher, (3) Interpreted, which represent what is described by professional participants who aim to make sure that researchers professionally see the data, and (4) Vaguing it out, which represent vague information provided by participants that do not bother to provide information to the researcher [39]. Any researcher can encounter any of these types of collected data.

However, conducting semi-structured open-ended guide allows the researcher to let the participants provide any detailed data and examples [80]. Also, semi-structured interviews help researchers in asking the same question in multiple ways throughout the interview protocol [80]. In this research study, each conducted interview was recorded and then transcribed verbatim for detailed analysis in a systematic continuous basis. Since most of the conducted interviews were carried out in Swedish, the quotations, which support the findings of this research study, were translated carefully into English.

Furthermore, this research encompasses conducting a longitudinal embedded case study, which involves a repeated process of observation of the same variables for a long period – around two years. This triangulation in data collection and analysis validates that derived data from observations did not contradict, but rather supported, the interview data.

Transferability refers to the extent to which the findings from one context are applicable to another while understanding the circumstances that affect the studied context [68]. A limitation of this research study is being limited to the context of a single case study. Case studies put more emphasis on recognising a contemporary phenomenon rather than generalising the findings. Also, the Grounded Theory does not claim producing a universally applicable theory, but it can be modified as new data from other contexts are constantly compared with the theory [13, 40]. Nevertheless, the interviews were conducted with different roles (such as Agile Coach, Product Owner, Key Account Manager, Chapter Leader, Senior Developer, Support Manager, etc.) from different squads to brace the external validity. Also, the findings of such qualitative research may benefit other companies in similar contexts.

Confirmability examines the researcher's objectivity in relation to the studies context [68]. In both of the longitudinal and intervention case studies, the interview protocols were tested through two pilot interviews and subsequently revised. Also, this research study adopted a grounded theory method, which is a systematic research methodology aims to generate an abstract theory through methodical gathering and analysis of data on a continuous basis [13]. Furthermore, a longitudinal and intervention embedded case studies were conducted during which the same variables were observed repeatedly. This triangulation, in turn, helped the researcher in finding any suspected deviation between "semi-structured interviews" view of matters and the "real" case [89].

9.4 Summary

In this chapter, the findings were analysed and interpreted in light of related literature. Also, this chapter has discussed the limitations of this research study.

Firstly, this research investigated the conflicting trade-offs between alignment and autonomy. A tension between squads autonomy and alignment was identified. The conflicting trade-offs are resolved by identifying influential factors on 3 different aspects of Spotify Tailoring. These aspects are (1) establishing and building autonomous squads, (2) aligning autonomous squads, and (3) performing B2B product development by tailoring the Spotify model. The discussion demonstrates that appropriate utilisation of these factors and their related identified practices can aid Agile practitioners in getting the balance right between squads autonomy and alignment.

Secondly, this research explored the tailoring approach used in FinTech's Spotify model and revealed a novel approach, called Heterogeneous Tailoring. The tailoring approaches identified by previous research compared to the Heterogeneous Tailoring approach. The analysis and interpretation show that the Heterogeneous Tailoring approach differs from other identified tailoring approaches in previous literature. The discussion presented in this chapter compares the aims of each tailoring approach, its advantages and disadvantages.

Thirdly, the exploration of Spotify tailoring in FinTechOrg identified a challenge in governing architectural decisions across heterogeneous autonomous squads. Hence, this research study introduced an approach to overcome this challenge. The proposed approach incorporates a structural change and an architecture change management process. Based on the results of the intervention conducted, the Heterogeneous Tailoring approach was adapted to accommodate the introduced approach to architectural governance. The discussion presented in this chapter compares and discusses the introduced approach to architecture governance to other approaches identified in previous literature.

This chapter has also presented and discussed the limitations of my research results. This research has employed a qualitative research design by conducting a single case study, which in turn affected the generalisability of this research study negatively. However, this research study involves a longitudinal embedded case study, which puts more emphasis on having an in-depth understanding of contemporary phenomenon than the generalisability of the findings. This in-depth understanding is especially important since there was a lack of scientific research on the Spotify model.

Chapter 10

Conclusion

10.1 Introduction

This research study was motivated by the existence of conflicting trade-offs between alignment and autonomy in the Spotify model. This model initiates the creation of autonomous yet aligned squads by employing an adaptive structure [58, 57]. Autonomy is important to have motivated and innovative squads that can deliver the developed software quickly to the customers [57, 58, 76]. The alignment of autonomous squads works as enabler for squads autonomy [57, 58]. However, there is a conflicting trade-off between autonomy and alignment. Too much alignment might hinder squad autonomy, but without alignment, the squads are autonomous yet not effective.

Also, the Spotify model does not provide guidelines for building squads autonomy nor for aligning autonomous squads. There is a lack of scientific research on the Spotify model. At the time this research study started in 2015, there was no scientific research on the Spotify model. Previous research on the Spotify model was limited to the grey literature [56, 57, 58, 69]. In 2017, however, Mankins and Garton [71] identified 3 challenges to getting the balance right between individual autonomy and organisational goals when using the Spotify model. Also, in 2019, Šmite et al., [106]

studied knowledge sharing in the Spotify organisation. Hence, this research aimed to (1) explore how Agile practitioners, from a FinTech organisation, resolve the conflicting trade-offs between squads autonomy and alignment, and (2) develop and evaluate new architectural governance practices, in a FinTech organisation tailoring the Spotify model.

This research study has employed *qualitative research design* to begin to fill the literature gaps identified about Spotify Tailoring in the FinTech industry of large-scale agile. In essence, the employed research design consists of a longitudinal embedded case study and a practical intervention embedded case study. Data were collected through semi-structured open-ended interviews and supplemented by direct observation, documentation, and artefacts. The collected data was analysed using an approach informed by the Grounded Theory.

Five main contributions were presented in this thesis, which are summarised Sect. 10.3. Fig. 1.1 illustrates the connections between the research questions, contributions, and conducted scientific research by this study [93, 94, 95, 96, 97, 98, 99]. The contributions of this research study are spread through Chapter 4-8.

This chapter consolidates all the previous chapters by providing an overview of all chapters. This overview is followed by a summary of the contributions of this research study. Then, a reflection on this research study, by discussing some encountered implications, is provided. Finally, this chapter presents suggestions for future work through which this research study can be extended.

10.2 Summary of the Thesis

The introduction chapter showed the importance of addressing the lack of scientific research on the Spotify model. This research study aimed to resolve the conflicting trade-offs between autonomy and alignment and to develop and evaluate new architectural governance practices by adapting the Spotify model. The introduction chapter also showed the connection between the research questions, contributions, and conducted scientific research by this research.

In Chapter 2, a literature review was conducted, which comprises a minor upfront literature review and a detailed literature review. The minor literature review was conducted to understand the basic facts and terminologies of the Spotify model and the Agile architecture to converse with the participants during interviews. A detailed literature review was conducted based on the grounded findings to facilitate the research discussion. Thus, an overview of the Spotify model and some related aspects to the findings were presented. For example, Chapter 2 presents some Agile methods (i.e., Lean and Scrum), Agile method tailoring approaches, large-scale agile, inter-team coordination, autonomous teams, and Agile architecture.

In Chapter 3, the adopted research approach was described by following Creswells' framework. This research utilises *qualitative research design*. This design comprises a longitudinal embedded case study and embedded case study intervention. The longitudinal embedded case study lasted over 21 months, during which 225 ceremonies were observed, 14 semi-structured open-ended interviews were conducted, and data were collected from different sorts of artefacts. The collected data from the longitudinal part was analysed using an approach informed by the Grounded Theory. The intervention embedded case study lasted over 3 months, during which 32 ceremonies were observed and 8 semi-structured open-ended interviews were conducted. The collected data was analysed using an approach informed by the Grounded Theory.

Chapters 4-7 describe the outcome of the conducted longitudinal embedded case study. This part is followed by an intervention embedded case study to tackle one of the identified challenges. Chapter 8 describe the outcome of the intervention part and its embedded case study.

In Chapter 4, the following research question was addressed: *How do Agile practitioners, from a FinTech organisation, establish and build squad autonomy?* This chapter identified influential factors on establishing and building autonomous squads. These factors are (1) tailoring the adaptive structure and creating sufficient communities around it, (2) building squad or mission-based strategy, and (3) tailoring Agile methods on the squad or mission level.

In Chapter 5, the following research question was addressed: *How do Agile practition-*

ers, from a FinTech organisation, achieve and sustain the alignment of autonomous squads? This research identified influential factors on aligning autonomous squads. The investigation of Spotify Tailoring, from a squad alignment perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. Thus, 31 employed practices and attributes were identified. These practices and attributes were classified into 5 modified, 12 new, and 14 previously introduced practices and attributes in the Spotify model.

Also, Chapter 5 identified 7 influential factors on aligning autonomous squads. These influential factors are; (1) adaptive structure, (2) collective code ownership, (3) collective decision-making, (4) knowledge sharing, (5) inter-team coordination, (6) mission-based planning, and (7) delivery strategy. Each factor is supported by a set of practices and attributes that can strengthen the alignment of autonomous squads. The identified influential factors and their related practices can aid Agile practitioners in aligning autonomous squads.

In Chapter 6, the following research question was addressed: *How do Agile practitioners, from a FinTech organisation, tailor the Spotify model for B2B product development?* This chapter revealed two main aspects related to product development using the Spotify model. Firstly, this chapter has identified the impact of product development on squads autonomy and alignment. This impact is mainly realised in the identification of squads' missions strategy and the alignment of product backlog items among autonomous squads.

Secondly, Chapter 6 identified aspects related to performing B2B product development by tailoring the Spotify model. The investigation of Spotify Tailoring, from a B2B perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. Thus, *44* tailored practices and attributes were identified. These practices and attributes were classified into 6 modified, 28 new, and 10 previously introduced practices and attributes in the Spotify model. Also, Chapter 6 identified influential factors on Spotify tailoring for B2B product development. These influential factors are (1) project visibility, (2) interactions within product development, (3) building relationships, and (4) response time. Each identified factor is supported by a set of practices and attributes that facilitate B2B product development. The identified factors and their

related practices can aid Agile practitioners in performing B2B product development by tailoring the Spotify model.

In Chapter 7, the following research question was addressed: *What is the approach taken to Agile tailoring, when using the Spotify model?* This chapter revealed a novel approach to Agile tailoring, called *"Heterogeneous Tailoring"*. Three key features characterise the Heterogeneous Tailoring approach. Firstly, each autonomous squad is empowered to select and tailor its development method. Secondly, each squad is aligned with other squads and common product development goals. Thirdly, product steering committee draws the strategy of squads missions and aligns the product backlog among autonomous squads.

Also, Chapter 7 identified some benefits for the Heterogeneous Tailoring approach. These benefits include: (1) mitigating challenges of utilising Agile practices across squads that do not fit the needs of each squad, (2) improved creativity or productivity, (3) and mitigated risks of divergence from shared development objectives by employing alignment practices. However, this research identified two challenges to this approach. Firstly, it is difficult to measure the overall code quality, performance and productivity for different squads. Secondly, aligning and governing enterprise architectural decisions across autonomous squads requires new practices

Chapter 8 addressed the following research question: *What new architectural governance practices can be introduced for loosely coupled yet cooperating squads?* An intervention embedded case study was conducted to minimise delays in architecture based decision making, level up the quality of software being produced, and overcome obstacles to aligning and governing architectural decisions across autonomous squads. This intervention introduced a novel approach to architectural governance using the Spotify model, which incorporates a structural change and an architecture change management process. Chapter 8 presented the characteristics of the introduced approach to architectural governance, its benefits, and identify guidelines and challenges for those wishing to adopt it. In addition, Chapter 8 presented how the Heterogeneous Tailoring approach was adapted based on this intervention.

Chapter 9 discussed the findings from Chapters 4-8 based on the literature review in

Chapter 2. Also, Chapter 9 discussed the limitations of the research study.

10.3 Contributions

The main contributions of this research are the following:

- **Contribution 1:** Influential factors on establishing and building autonomous squads were identified. These factors consist of: (1) tailoring the adaptive structure and creating communities around it, (2) building squad or mission-based strategy, and (3) tailoring Agile methods on the squad or mission level.
- **Contribution 2:** Influential factors on aligning autonomous squads were identified. My investigation of Spotify Tailoring, from a squad alignment perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. Also, my investigation has identified influential factors on aligning autonomous squads. Each identified factor is supported by a set of practices and attributes that can strengthen the alignment of autonomous squads. The identified influential factors and their related practices can aid Agile practitioners in aligning autonomous squads.
- **Contribution 3:** Two main aspects related to product development were identified. Firstly, the impact of product development on squads autonomy and alignment was identified. This impact is mainly realised in the identification of squads' missions strategy and the alignment of product backlog items among autonomous squads. Secondly, performing B2B product development by tailoring the Spotify model was identified. My investigation of Spotify Tailoring, from a B2B perspective, identified new, modified and previously introduced practices and attributes to the Spotify model. Also, my investigation has identified influential factors on Spotify tailoring for B2B product development. Each identified factor is supported by a set of practices and attributes that facilitate B2B FinTech product development. The identified influential factors and their related practices facilitate performing B2B product development using the Spotify model.

Note: The Contributions 1-3 facilitate resolving the conflicting trade-offs between autonomy and alignment. The autonomy of squads improves the motivation and innovation, which in turn speeds up the release delivery process. Also, the alignment of autonomous squads works as enable for squad autonomy. This research has identified a tension between squads autonomy and alignment. Too much alignment might hinder squad autonomy, but at the same time without alignment, the squads can be autonomous yet not effective. Therefore, Agile practitioners should get the balance right between squads autonomy and alignment.

- **Contribution 4:** A novel approach to Agile tailoring was revealed, called Heterogeneous Tailoring. Three key features characterise this approach. Firstly, each autonomous squad is empowered to select and tailor its development method. Secondly, each squad is aligned with other squads and to common product development goals and objectives. Thirdly, product steering committee draws the strategy of squads' missions and aligns the product backlog among autonomous squads.
- **Contribution 5:** A novel approach to architectural governance, by tailoring the Spotify model, was develop and evaluate. This architectural governance approach incorporates an organisational structural change and an architecture change management process. The impact of the architectural governance approach on the Heterogeneous Tailoring approach was identified. Also, the Heterogeneous Tailoring approach was adapted to accommodate the newly introduced approach to architectural governance.

10.4 Reflection

Creating an initial research design, identifying valid research question, and aims and objectives were the most significant challenges during this research. Initially, this research aimed to introduce a holistic approach to Agile tailoring. To verify the approach to Agile tailoring, a quantitative research design (i.e., survey) was proposed. This PhD proposal was submitted and accepted.

However, the researcher was struggling to employ the quantitative design to verify and validate the proposed approach, which consists of a wide range of dimensions and attributes. The huge scope of the initial research and the challenge of applying the quantitative design to verify and validate the proposed approach made it difficult to proceed in this direction. Also, the original supervisor moved to industry, in the middle of this chaotic period, which resulted in a change of the supervisor and hence shifted the focus of this research into its current state.

I needed to be self-motivated and develop an intense curiosity about the research problem. Identifying a reasonable scope of a research study that can be conducted within a specific time frame is vital. The process involved in the early stage of research, if planned and done correctly, will help researchers to rethink their research aim and objectives, and establish a foundation for rigour quite early in the PhD study.

Embedding the researcher in the case study added breadth and depth to data collection. The collected data in this research study includes observations, interviews, documentation, and artefacts of different kinds – such as product backlog, tools, and technology. Hence, conducting an embedded case study brought a richness of data through triangulation [91]. This triangulation of data collection, in turn, ensured the authenticity of the data collected through interviews and contributed to the validity of this research, where the interpretation of the interview data was compared to the observations of the workplace and activities. Despite the tremendous benefits of this triangulation process, it can be expensive and time consuming for PhD students.

The Grounded Theory method does not claim to produce a universally applicable theory [40]. The Grounded Theory can be modified by constant comparison to further collected data from new contexts (i.e., other software industries, projects, organisations, and cultures). To the best of my knowledge, the grounded theory of Heterogeneous Tailoring is the first of its kind in the field of Software Engineering. Conducting multiple case studies in software organisations using the Spotify model will help in theory testing in the sense of performing statistical generalisation to a large population, which was not the aim of my research.

The researcher had planned with the new supervisor to consider and think of the published

scientific papers as a skeleton of the core chapters of the thesis. These publications can then be expanded into sections or chapters in the final thesis. This strategy was proved to be useful for me. However, this approach could introduce a potential risk of underestimating the remaining work for thesis write-up. Re-writing and arranging already published ideas into appropriate sections and chapters can be time-consuming. Though, the researcher will not start the writing-up on blank pages since the ideas exist and already published. This process increases the researcher's preparation and confidence to defend the research.

Being a part-time PhD student was challenging. I have chosen this mode of study because of other essential responsibilities in my life, including family and job. Balancing different roles in life resulted in a limited spent time at the university, mainly when the researcher is located offshore. However, the constant commitment and support of the supervisor and researcher's ability to being resilient, motivated, passionate, enthusiastic, and organised throughout this journey were crucial to complete this research.

10.5 Future work

This research has begun to fill the literature gaps identified about Spotify Tailoring in FinTech organisations of large-scale. The researcher started approaching the identified research gaps by exploring the conflicting trade-offs between squads autonomy and alignment identifying influential factors on different aspects related to the Spotify Tailoring. Yet, this research can be extended by considering other settings (i.e., other software industries, projects, organisations, and cultures). Exploring how Spotify Tailoring is achieved in different contexts will enrich literature and improve the generalisability of the identified influential factors on building autonomous squads and aligning them, which can facilitate the identification of guidelines for building squads autonomy and alignment.

Further research can also be conducted to investigate issues surrounding the Heterogeneous Tailoring approach. These issues include measuring the overall code quality,

performance and productivity for different autonomous squads. In the Heterogeneous Tailoring approach, each squad or a mission considers its way of measuring code quality, progress, velocity, and success. Hence, each squad or mission has its key performance indicators. These indicators are used to produce quantifiable measures to evaluate the success of each squad. Integrating all used key performance indicators for all squads or missions is challenging.

Agile practitioners should get the balance right between squads autonomy and alignment. Hence, future research can provide key indicators to achieve a seemingly paradoxical combination of squads autonomy and alignment and balance them. Such key indicators can be employed in a maturity model or diagnostic tool to help organisations in understanding their current practices and work toward improving their agility. These maturity models or diagnostic tools can guide organisations toward reaching different levels of balance between squads autonomy and alignment.

References

[1] N. Abbas, A. M. Gravell, and G. B. Wills. Using factor analysis to generate clusters of agile practices (a guide for agile process improvement). In *2010 Agile Conference*, pages 11–20, 2010.

[2] P. Abrahamsson, M. A. Babar, and P. Kruchten. Agility and architecture: Can they coexist? *IEEE Software*, 27(2):16–22, March 2010.

[3] E. Ahmed and A. Sidky. 25 percent ahead of schedule and just at "step 2 of the sami. In *Proceedings of the 2009 Agile Conference*, AGILE 09, page 162169, USA, 2009. IEEE Computer Society.

[4] C. Anderson and E. McMillan. Of ants and men: Self-organized teams in human and insect organizations. *Emergence: Complexity and Organization*, 5(2):29–41, 2003.

[5] L. Andrews, A. Higgins, M. W. Andrews, and J. G. Lalor. Classic grounded theory to analyse secondary data: Reality and reflections. *Grounded Theory Review: An International Journal*, 11(1), 2012.

[6] H. Ayed, B. Vanderose, and N. Habra. A metamodel-based approach for customizing and assessing agile methods. In *2012 Eighth International Conference on the Quality of Information and Communications Technology*, pages 66–74, 2012.

[7] M. A. Babar and P. Abrahamsson. Architecture-centric methods and agile approaches. In Pekka Abrahamsson, Richard Baskerville, Kieran Conboy, Brian Fitzgerald, Lorraine Morgan, and Xiaofeng Wang, editors, *Agile Processes in Soft-*

ware Engineering and Extreme Programming, pages 242–243, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[8] J. M. Bass. Agile method tailoring in distributed enterprises: Product owner teams. In *2013 IEEE 8th International Conference on Global Software Engineering*, pages 154–163, Aug 2013.

[9] J. M. Bass. How product owner teams scale agile methods to large distributed enterprises. *Empirical Software Engineering*, 20(6):1525–1557, December 2015.

[10] J. M. Bass and A. Haxby. Tailoring product ownership in large-scale agile projects: Managing scale, distance, and governance. *IEEE Software*, 36(2):58–63, March 2019.

[11] S. Bellomo, P. Kruchten, R. L. Nord, and I. Ozkaya. How to agilely architect an agile architecture. *Cutter IT Journal*, 27(2):12–17, 2014.

[12] S. Bick, A. Scheerer, and K. Spohrer. Inter-team coordination in large agile software development settings: Five ways of practicing agile at scale. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, XP '16 Workshops, USA, 2016.

[13] D. F. Birks, W. Fernandez, N. Levina, and S. Nasirin. Grounded theory method in information systems research: its nature, diversity and opportunities. *European Journal of Information Systems*, 22(1):1–8, 2013.

[14] S. Blank. Why the lean start-up changes everything. *Harvard business review*, 91 (5):63–72, 2013.

[15] G. Booch. The defenestration of superfluous architectural accoutrements. *IEEE Software*, 26(4):7–8, July 2009.

[16] F. Buschmann and K. Henney. Architecture and agility: Married, divorced, or just good friends? *IEEE Software*, 30(2):80–82, March 2013.

[17] A. S. Campanelli and F. S. Parreiras. Agile methods tailoring a systematic literature review. *Journal of Systems and Software*, 110:85 – 100, 2015.

[18] A. S. Campanelli, R. D. Camilo, and F. S. Parreiras. The impact of tailoring criteria on agile practices adoption: A survey with novice agile practitioners in brazil. *Journal of Systems and Software*, 137:366 – 379, 2018.

[19] S. Casare, T. Ziadi, A. A. F. Brando, and Z. Guessoum. Meduse: An approach for tailoring software development process. In *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 197–200, 2016.

[20] T. Chow and D. Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961 – 971, 2008. Agile Product Line Engineering.

[21] A. Cockburn. *Agile software development: the cooperative game*. Pearson Education, 2006.

[22] A. Cockburn and J. Highsmith. Agile software development, the people factor. *Computer*, 34(11):131–133, Nov 2001.

[23] K. Conboy and N. Carroll. Implementing large-scale agile frameworks: Challenges and recommendations. *IEEE Software*, 36(2):44–50, March 2019.

[24] K. Conboy and B. Fitzgerald. Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion. *ACM Trans. Softw. Eng. Methodol.*, 20(1):2:1–2:30, July 2010.

[25] J. W. Creswell and J. D. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications Ltd., 5 edition, 2018.

[26] K. Dikert, M. Paasivaara, and C. Lassenius. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119:87 – 108, 2016.

[27] T. Dingsøyr, T. Dybå, and N. B. Moe. *Agile Software Development: Current Research and Future Directions*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[28] T. Dingsøyr, T. E. Fægri, and J. Itkonen. What is large in large-scale? a taxonomy of scale for agile software development. In A. Jedlitschka, P. Kuvaja,

M. Kuhrmann, T. Männistö, J. Münch, and M. Raatikainen, editors, *Product-Focused Software Process Improvement*, pages 273–276, Cham, 2014. Springer International Publishing.

[29] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim. Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*, 23(1):490–520, February 2018.

[30] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9):833 – 859, 2008.

[31] H. Erdogmus. Architecture meets agility. *IEEE Softw.*, 26(5):2–4, September 2009.

[32] H. C. Esfahani, S. Eric, and M. C. Annosi. Towards the strategic analysis of agile practices. In *CAiSE Forum*, pages 155–162, 2011.

[33] D. Falessi, G. Cantone, S. A. Sarcia', G. Calavaro, P. Subiaco, and C. D'Amore. Peaceful coexistence: Agile developer perspectives on software architecture. *IEEE Software*, 27(2):23–25, March 2010.

[34] B. Fitzgerald and K. Stol. Continuous software engineering: A roadmap and agenda. *The Journal of Systems & Software*, 123:176–189, 2017.

[35] B. Fitzgerald, N. L. Russo, and T. O'Kane. An empirical study of system development method tailoring in practice. In *ECIS*, 2000.

[36] B. Fitzgerald, N. L. Russo, and T. O'Kane. Software development method tailoring at motorola. *Commun. ACM*, 46(4):64–70, April 2003.

[37] M. Fowler and J. Highsmith. The agile manifesto, 2001. URL http://www.agilemanifesto.org/.

[38] A. Q. Gill, B. Henderson-Sellers, and M. Niazi. Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. *Information Systems Frontiers*, 20(2):315341, April 2018.

[39] B. G. Glaser. *Theoretical sensitivity: advances in the methodology of grounded theory / Barney G. Glaser*. Sociology Press Mill Valley, Calif, 1978.

[40] B. G. Glaser. *Doing grounded theory: Issues and discussions*. Sociology Press, 1998.

[41] B. G. Glaser and A. L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter, New York, NY, 1967.

[42] F. González, L. Silvestre, M. C. Bastarrica, and M. Solari. Template-based vs. automatic process tailoring. In *2014 33rd International Conference of the Chilean Computer Science Society (SCCC)*, pages 124–127, 2014.

[43] P. Gregory and K. Taylor. Defining agile culture: A collaborative and practitioner-led approach. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 37–38, May 2019.

[44] M. A. Hart. The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses eric ries. *Journal of Product Innovation Management*, 29(3):508–509, 2012.

[45] T. Hayata and J. Han. A hybrid model for it project with scrum. In *Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics*, pages 285–290, 2011.

[46] B. Henderson-Sellers and J. Ralyté. Situational method engineering: State-of-the-art review. *Journal of Universal Computer Science*, 16(3):424–478, 2010. cited By 129.

[47] J. A. Highsmith. *Agile project management : creating innovative products*. Addison-Wesley, 2004.

[48] R. Hoda, J. Noble, and S. Marshall. Organizing self-organizing teams. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 285–294, New York, NY, USA, 2010. ACM.

[49] R. Hoda, J. Noble, and S. Marshall. Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering*, 17(6): 609–639, Dec 2012.

[50] R. Hoda, J. Noble, and S. Marshall. Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering*, 39(3):422–444, March 2013.

[51] R. Hoda, N. Salleh, J. Grundy, and H. M. Tee. Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology*, 85:60 – 70, 2017.

[52] T. Imani, M. Nakano, and V. Anantatmula. Does a hybrid approach of agile and plan-driven methods work better for it system development projects? *International journal of engineering research and applications*, 1(2):3, 2017.

[53] S. Jalali and C. Wohlin. Agile practices in global software engineering - a systematic map. In *2010 5th IEEE International Conference on Global Software Engineering*, pages 45–54, 2010.

[54] G. Kalus and M. Kuhrmann. Criteria for software process tailoring: A systematic review. In *Proceedings of the 2013 International Conference on Software and System Process*, ICSSP 2013, pages 171–180, New York, NY, USA, 2013. ACM.

[55] E. Kilu, F. Milani, E. Scott, and D. Pfahl. Agile software process improvement by learning from financial and fintech companies: Lhv bank case study. In D. Winkler, S. Biffl, and J. Bergsmann, editors, *Software Quality: The Complexity and Challenges of Software Engineering and Software Quality in the Cloud*, pages 57–69, Cham, 2019. Springer International Publishing.

[56] H. Kniberg. Spotify squad framework - part ii, April, 2014. URL https://medium.com/project-management-learnings/spotify-squad-framework-part-ii-c5d4b9398c30.

[57] H. Kniberg. Spotify squad framework - part i, January, 2014. URL https://medium.com/project-management-learnings/spotify-squad-framework-part-i-8f74bcfcd761.

[58] H. Kniberg and A. Ivarsson. Scaling agile spotify with tribes, squads, chapters & guilds, October, 2012. URL https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf.

[59] R. E. Kraut and L. A. Streeter. Coordination in software development. *Commun. ACM*, 38(3):69–81, March 1995.

[60] A. Krijnen. The toyota way: 14 management principles from the world's greatest manufacturer, 2007.

[61] P. Kruchten. Software architecture and agile software development: a clash of two cultures? In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 497–498, May 2010.

[62] N. Kurapati, V. S. C. Manyam, and K. Petersen. Agile software development practice adoption survey. In C. Wohlin, editor, *Agile Processes in Software Engineering and Extreme Programming*, pages 16–30, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[63] R. Kuusela and M. Koivuluoma. Lean transformation framework for software intensive companies: Responding to challenges created by the cloud. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 378–382. IEEE, 2011.

[64] C. Larman and B. Vodde. *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Addison-Wesley Professional, 1st edition, 2010.

[65] C. Larman and B. Vodde. Scaling agile development. *CrossTalk*, 9:8–12, 2013.

[66] D. Leffingwell. *Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series)*. Addison-Wesley Professional, 2007.

[67] D. Leffingwell. *SAFe 4.0 Reference Guide: Scaled Agile Framework for Lean Software and Systems Engineering*. Addison-Wesley Professional, 1st edition, 2016.

[68] Y. S. Lincoln and E. G. Guba. Naturalistic inquiry (vol. 75), 1985.

[69] B. Linders. Don't copy the spotify model, 2016. URL https://www.infoq.com/news/2016/10/no-spotify-model.

[70] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kahkonen. Agile software development in large organizations. *Computer*, 37(12):26–34, Dec 2004.

[71] M. Mankins and E. Garton. How spotify balances employee autonomy and accountability. *Harvard Business Review*, 95(1), 2017.

[72] V. S. C. Manyam and N. Kurapati. Empirical investigation on adoption and adaptation of agile practices. Master's thesis, Blekinge Institute of Technology, 2012.

[73] A. Martini and J. Bosch. A multiple case study of continuous architecting in large agile companies: Current gaps and the caffea framework. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 1–10, 2016.

[74] C. Melo, D. Cruzes, F. Kon, and R. Conradi. Interpretative case studies on agile team productivity and management. *Information and Software Technology*, 2013.

[75] P. Middleton and J. Sutton. *Lean software strategies: proven techniques for managers and developers*. CRC Press, 2005.

[76] N. B. Moe and T. Dingsøyr. Scrum and team effectiveness: Theory and practice. In P. Abrahamsson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan, and X. Wang, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 11–20, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[77] N. B. Moe and T. Dingsøyr. Emerging research themes and updated research agenda for large-scale agile development: A summary of the 5th international workshop at xp2017. In *Proceedings of the XP2017 Scientific Workshops*, XP 17, New York, NY, USA, 2017. Association for Computing Machinery.

[78] N. B. Moe, T. Dingsøyr, and T. Dybå. Understanding self-organizing teams in agile software development. In *19th Australian Conference on Software Engineering (aswec 2008)*, pages 76–85, March 2008.

[79] N. B. Moe, H. H. Olsson, and T. Dingsøyr. Trends in large-scale agile development: A summary of the 4th workshop at xp2016. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, page 1. ACM, 2016.

[80] M.D. Myers and M Newman. The qualitative interview in is research: Examining the craft. *Information and Organization*, 17(1):2 – 26, 2007.

[81] S. Nerur. Acceptance of software process innovations the case of extreme programming. *European Journal of Information Systems*, 18(4):344–354, 2009.

[82] R. L. Nord, I. Ozkaya, and P. Kruchten. Agile in distress: Architecture to the rescue. In T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen, editors, *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, pages 43–57, Cham, 2014. Springer International Publishing.

[83] B. J. Oates. *Researching Information Systems and Computing*. Sage Publications Ltd., 2006.

[84] N. Ozkan and A. Tarhan. A review of scaling approaches to agile software development models. *Software Quality Professional*, 21(4):11–20, 09 2019. Name - Spotify AB; Copyright - Copyright American Society for Quality Sep 2019; Last updated - 2019-11-06; SubjectsTermNotLitGenreText - New York.

[85] M. Paasivaara, S. Durasiewicz, and C. Lassenius. Using scrum in a globally distributed project: A case study. *Softw. Process*, 13(6):527–544, November 2008. ISSN 1077-4866.

[86] M. Paasivaara, C. Lassenius, and V. T. Heikkilä. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 235–238, Sept 2012.

[87] K. W. Parry. Grounded theory and social process: A new direction for leadership research. *The Leadership Quarterly*, 9(1):85 – 105, 1998.

[88] J. Porter. Spotify is first to 100 million paid subscribers, 2019. URL https://www.theverge.com/2019/4/29/18522297/.

[89] H. Robinson, J. Segal, and H. Sharp. Ethnographically-informed empirical studies of software practice. *Information and Software Technology*, 49(6):540 – 551, 2007.

[90] K. H. Rolland, B. Fitzgerald, T. Dingsøyr, and K. Stol. Problematizing agile in the large: alternative assumptions for large-scale agile development. In *ICIS 2016 PROCEEDINGS : 37 International Conference on Information Systems*, 2016.

[91] P. Runeson, M. Host, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley Publishing, 2012.

[92] A. Salameh. On process tailoring-an agile example. Master's thesis, Chalmers University of Technology, 2011.

[93] A. Salameh and J. M. Bass. Influential factors of aligning spotify squads in mission-critical and offshore projects – a longitudinal embedded case study. In M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, and S. Küpper, editors, *Product-Focused Software Process Improvement*, pages 199–215, Cham, 2018. Springer International Publishing.

[94] A. Salameh and J. M. Bass. Spotify tailoring for promoting effectiveness in cross-functional autonomous squads. In R. Hoda, editor, *Agile Processes in Software Engineering and Extreme Programming – Workshops*, pages 20–28, Cham, 2019. Springer International Publishing.

[95] A. Salameh and J. M. Bass. Spotify tailoring for B2B product development. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 61–65, Aug 2019.

[96] A. Salameh and J. M. Bass. An architectural governance approach by tailoring the spotify model. *AI & Society: Journal of Knowledge, Culture and Communication*, 2020. Special issue on team autonomy in digital transformations. Submitted.

[97] A. Salameh and J. M. Bass. Spotify tailoring for architectural governance. In *Agile Processes in Software Engineering and Extreme Programming – Workshops*, Cham, 2020. Springer International Publishing. Accepted.

[98] A. Salameh and J. M. Bass. Heterogeneous tailoring approach using the spotify model. In *Proceedings of the Evaluation and Assessment in Software Engineering*, EASE 20, page 293298, New York, NY, USA, 2020. Association for Computing Machinery.

[99] A. Salameh and O. Jaradat. A safety-centric change management framework by tailoring agile and v-model processes. In *36th International System Safety Conference ISSC 2018, 13 Aug 2018, Phoenix, AZ, United States*, 2018.

[100] T. Sauvola, L. E. Lwakatare, T. Karvonen, P. Kuvaja, H. H. Olsson, J. Bosch, and M. Oivo. Towards customer-centric software development: A multiple-case study. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 9–17, Aug 2015.

[101] A. Scheerer, T. Hildenbrand, and T. Kude. Coordination in large-scale agile software development: A multiteam systems perspective. In *2014 47th Hawaii International Conference on System Sciences*, pages 4780–4788, Jan 2014.

[102] K. Schwaber. *Agile project management with Scrum*. Microsoft Press, Redmond, Wash., 2004.

[103] A. W. Scott and M. Lines. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press, 1st edition, 2012.

[104] P. Serrador and J. K. Pinto. Does agile work? - a quantitative analysis of agile project success. *International Journal of Project Management*, 33(5):1040–1051, 2015.

[105] D. Šmite, C. Wohlin, Z. Galviņa, and R. Prikladnicki. An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering*, 19(1):105–153, Feb 2014.

[106] D. Šmite, N. B. Moe, G. Levinta, and M. Floryan. Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizations. *IEEE Software*, 36(2): 51–57, March 2019.

[107] S. Soundararajan, J. D. Arthur, and O. Balci. A methodology for assessing agile software development methods. In *2012 Agile Conference*, pages 51–54, 2012.

[108] M. Staron. *Action Research in Software Engineering*. Springer, 2020.

[109] A. Strauss and J. Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage, Newbury Park, California, 1990.

[110] V. Stray, N. B. Moe, and R. Hoda. Autonomous agile teams: Challenges and future directions for research. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*, XP '18, pages 16:1–16:5, New York, NY, USA, 2018. ACM.

[111] A. Sutharshan and S. P. Maj. An evaluation of agile software methodology techniques. *International Journal of Computer Science and Network Security*, 10 (12), 2010.

[112] H. Takeuchi and I. Nonaka. The new newman product development game. *Harvard Business Review*, 64(1):137–146, 1986.

[113] S. Theobald, A. Schmitt, and P. Diebold. Comparing scaling agile frameworks based on underlying practices. In R. Hoda, editor, *Agile Processes in Software Engineering and Extreme Programming – Workshops*, pages 88–96, Cham, 2019. Springer International Publishing.

[114] J. F. Tripp and D. J. Armstrong. Exploring the relationship between organizational adoption motives and the tailoring of agile methods. In *2014 47th Hawaii International Conference on System Sciences*, pages 4799–4806, 2014.

[115] J. F. Tripp and D. J. Armstrong. Agile methodologies: Organizational adoption motives, tailoring, and performance. *Journal of Computer Information Systems*, 58(2):170–179, 2018.

[116] D. Šmite, N. B. Moe, M. Floryan, G. Levinta, and P. Chatzipetrou. Spotify guilds. *Commun. ACM*, 63(3):5661, February 2020.

[117] J. P. Womack, D. T. Jones, D. Roos, et al. The machine that changed the world: The story of lean production. *New York: Rawson Associates*, 85, 1990.

[118] C. Yang, P. Liang, and P. Avgeriou. A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software*, 111:157 – 184, 2016.

Appendix A

Approved Ethical Application and Documents

A.1 Ethical Approval Form

Research, Innovation and Academic Engagement Ethical Approval Panel

Research Centres Support Team
G0.3 Joule House
University of Salford
M5 4WT

T +44(0)161 295 5278

www.salford.ac.uk/

28 February 2017

Ref: Abdallah Salameh

Dear Abdallah,

RE: ETHICS APPLICATION ST 16/127 – Agile Method Tailoring in Software Development Teams within Service-Oriented Enterprises

Based on the information you provided, I am pleased to inform you that your application ST ST 16/127 has been approved.

If there are any changes to the project and/ or its methodology, please inform the Panel as soon as possible by contacting S&T-ResearchEthics@salford.ac.uk

Yours sincerely,

Prof Mohammed Arif
Chair of the Science & Technology Research Ethics Panel
Professor of Sustainability and Process Management,
School of Built Environment
University of Salford
Maxwell Building, The Crescent
Greater Manchester, UK M5 4WT
Phone: + 44 161 295 6829
Email: m.arif@salford.ac.uk

A.2 Consent for Participation in Research

Research Governance and Ethics Committee (RGEC): College of Computer Science, University of Salford

Research Participant Consent Form

Title of Project: Agile Method Tailoring in Software Development Teams within Service-Oriented Enterprises

RGEC Ref No: ST 16/127

➢ I confirm that I have read and understood the information given in the questionnaire form for the above study and what my contribution will be.	**Yes**	**No**	
➢ I confirm that I have read and understood the information given in regard to providing the researcher with a permission to observe the software development lifecycle of our team for the sake of the above study, and that I agree to be observed.	**Yes**	**No**	
➢ I have been given the opportunity to ask questions (face to face, via telephone and e-mail)	**Yes**	**No**	
➢ I agree to take part in the interview	**Yes**	**No**	**NA**
➢ I agree to the interview being tape recorded	**Yes**	**No**	**NA**
➢ I understand that my participation is voluntary and that I can withdraw from the research at any time **without giving any reason**	**Yes**	**No**	
➢ **I agree to take part in the above study**	**Yes**	**No**	

Name of participant / company ……

Signature ……

Date ……

Name of researcher taking consent ……

Researcher's e-mail address ……

Research Governance and Ethics Committee Consent Form
Version 2 (06.01.11)

Appendix B

Interview Guide

B.1 Longitudinal Embedded Case Study

The duration of this interview will be 45-60 minutes. We have divided the interview into 3 main sections based on the type of questions. Section 1 consists of short questions that require direct answers. Section 2 consists of questions regarding the used methodology in the organisation, section 3 consists of questions that require detail answers, as they will determine the results for the research. The last section is about extracting other details if there were any.

Section 1: Organisation

In this section, we will be asking questions regarding the organisation in order to have a general overview of the experience of the organisation in agile methodology. As this information will help us understand whether organisations of different experiences with agile software development face similar problems with communication, coordination, and planning among FinTech organisations or not.

1. What is the age of the company?
2. What is the project management structure? Describe it?
    (a) How do you describe the management style?

(b) How many people working on in the project?

(c) How many teams?

3. What is your organisation's experience with Agile software development?

Section 2: Methodology

In this section, we will start by asking the following question in order to get information about what methodology the organisation uses for developing the project(s).

4. What Agile software development methodology does your organisation use? Please explain me how you use it (the full lifecycle)?

   Based on the reply of the above question we will ask further questions such as the following in order to understand how the organisation uses Agile and whether they face any issue:

   ***** (IF there was offshore/distributed teams or customers)**

   (a) How does your organisation manage time difference between the teams?

   (b) Do the offshore and onshore teams use the same development methodology?

      i. How do you communicate and plan the work?

      ii. What practices are you using to facilitate the development process?

Section 3: Research

In this section, we ask for detail description of the following questions in order to understand if the organisation is facing or has faced planning, communication and coordination issues while developing the software offshore and if they managed to overcome it or not:

5. How do you do the sprint/iteration and release planning (explain it please)?

   (a) How many sprints/iterations do you have for each release?

   (b) What is an iteration/release length (week)?

      i. If it was a short one or unstable size: Why?

      ii. Is it stressful for the team?

   ***** Preparatory meeting**

(c) Is there a preparatory meeting or session prior to the Iteration Planning Meeting where decisions are also made?

i. Can you describe the purpose of this meeting?

ii. What type of decisions are taken?

A. Who make them?

B. What challenges do you have?

6. How decision-making is conducted in your team?

(a) What kind of decisions are important to facilitate your work?

(b) Who makes such decisions? Why?

7. How efficient is the current methodology in handling the planning, communication and coordination of work? Please explain.

***** Planning**

(a) Is your burndown chart being ideal where you always on schedule?

(b) IF not:

i. What obstacles do you encounter?

ii. What are the reasons behind the delay?

iii. What is the employed process for handling a delay in the schedule?

(c) How do you describe the productivity results of your team(s) using the current methodology?

(d) How do you describe the produced product quality?

i. What can be improved?

ii. How would you improve it?

***** Knowledge sharing**

(e) How knowledge sharing is handled?

i. What would improve?

8. What difficulties or problems have you encountered in your development methodology during the project?

(a) Have you handled them? How?

(b) What remains to be resolved?

9. What adaptation/change would help you improve your work (challenges)?

(a) Explain why?

(b) Explain how?

Section 4: Other comments

10. What else would you like to add or we didn't talk about where you think that it is important to mention?

B.2 Intervention Embedded Case Study

The duration of this interview will be around 30 minutes. We have divided the interview into 3 main sections. Section 1 consists of questions about the organisational structural changes. Section 2 consists of questions about the architecture change management process. The last section is about extracting other details if there were any.

Section 1: Structural Change

In this section, we will be asking questions about the proposed changes to the organisational structure to collect a general overview of the practitioner experience for this change.

1. How the structural change achieved better governance and alignment of architectural decisions?
   (a) What are the responsibilities of Architecture Owner?
   (b) What are the responsibilities of Enterprise Architect?
   ***** If the participant has Architecture Owner or Enterprise Architect role:**
   (c) Do your new responsibilities increase the pressure (overhead) on you? Please explain?
2. What did not work well with aligning and governing architectural decisions (challenges)? Why?
3. How would you improve aligning architectural decisions to be more effective?
4. What else would you like to add about the aligning architectural decisions?

Section 2: Change Management Process

In this section, we will be asking questions to get feedback about the employed change management process for aligning and governing architectural decisions.

5. What worked well with the change management process? Why?

   ***** If the participant has Architecture Owner role:**

   (a) How well Activity 2 (i.e., Understanding the change and its impact) worked? Why?

   (b) How well Activity 4 (i.e., Investigating the impacted components to decide on the required change) worked? Why?

   (c) How well Activity 5 (i.e., Deriving more user stories) worked? Why?

   ***** If the participant has Enterprise Architect role:**

   (d) How well Activity 3 (i.e., Understanding the change and identify the impacted components) worked? Why?

   ***** If the participant has Enterprise PO role:**

   (e) How well Activity 6 (i.e., Planning implementation of change) process worked? Why?

   ***** If the participant has a developer role:**

   (f) How well the processes of "implementation, testing, and delivery of the change request" worked (Activity 7-9)? Why?

6. What did not work well with the proposed change management process (challenges)? Why?

7. How would you improve or tailor the process to be more effective?

8. What else would you like to add about the change management process?

Section 3: Other comments

9. What else would you like to add or we didn?t talk about where you think that it is important to mention?