The Journal of Systems & Software 200 (2023) 111649

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

In practice

Decentralized decision-making and scaled autonomy at Spotify[☆]

Darja Šmite[a,*], Nils Brede Moe[b], Marcin Floryan[c], Javier Gonzalez-Huerta[a], Michael Dorner[a], Aivars Sablis[d]

[a] Blekinge Institute of Technology Karlskrona, Sweden
[b] SINTEF Trondheim, Norway
[c] Spotify Stockholm, Sweden
[d] SAF Tehnika JSC Riga, Latvia

ARTICLE INFO

Article history:
Received 5 April 2022
Received in revised form 13 December 2022
Accepted 13 February 2023
Available online 15 February 2023

Keywords:
Scaling agile
Scaled autonomy
Enabling constraints
Coordination
Large-scale software development
The Spotify model

ABSTRACT

While modern software companies strive to increase team autonomy to enable them to successfully operate the piece of software they develop and deploy, efficient ways to orchestrate the work of multiple autonomous teams working in parallel are still poorly understood. In this paper, we report how team autonomy is maintained at Spotify at scale, based on team retrospectives, interviews with team managers and archival analysis of corporate databases and work procedures. In particular, we describe how managerial authority is decentralized through various workgroups with collective authority, what compromises are made to team autonomy to ensure alignment and which team-related factors can further hinder autonomy. Our findings show that scaled autonomy at Spotify does not mean anarchy, or unlimited permissiveness. Instead, squads are expected to take responsibility for their work and coordinate, communicate and align their actions with others, and comply with a few enabling constraints. Further, squads take many decisions independently without management control or due to collective efforts that bypass formal boundary structures. Mechanisms and strategies that enable self-organization at Spotify are related to effective sharing of the codebase, achieving alignment, networking and knowledge sharing, and are described to guide other companies in their efforts to scale autonomy.

1. Introduction

As the size, complexity, and diversity of today's software continues to grow, and the pace of change increases, software companies seek new ways to orchestrate their activities efficiently. One of the trends that helps to deal with these challenges is de-bureaucratization, which marks the journey towards creating complex jobs within simple organizations instead of simple jobs within complex organizations (de Sitter et al., 1997). As a result, organizational decentralization has in the recent decade gained mainstream consideration (Hackman, 1986; Lee and Edmondson, 2017; Olsson and Bosch, 2016). The steps taken by the companies to organize less hierarchically range from a situational delegation of authority to the radical company-wide formally recognized elimination of authority-over relationship between the management and subordinates (Lee and Edmondson, 2017). However, a common understanding of how complex interdependent work can be accomplished effectively at scale in the absence of managerial authority remains scarce (Lee and Edmondson, 2017).

Attempts to decentralize decision-making result in the need to practically support autonomy on the operational level, which in software companies means that more autonomy is given to teams. How much autonomy shall be given under which circumstances is still an open debate (Moe et al., 2021). Traditional agilists claim that *team autonomy* means that teams should be allowed to decide **how** to best accomplish their work (Sutherland and Schwaber, 2013). To achieve this, teams need to take up responsibility and be accountable, monitor their performance, manage and improve ways of working, and actively search for missing knowledge, or in other words, self-manage (Hackman, 1986). Many argue that teams shall have a broader authority than just the control over their own process. In practice, teams need a mandate over planning their work, assuring work quality, recruiting and dismissing members, or even suggesting new product ideas (Hackman, 1986) and making decisions with economic consequences (Guzzo and Dickson, 1996). Research into team autonomy suggests that members of autonomous teams

☆ Editor: Marcos Kalinowski.
* Corresponding author.
E-mail addresses: Darja.Smite@bth.se (D. Šmite), Nils.B.Moe@sintef.no (N.B. Moe), MFloryan@spotify.com (M. Floryan), Javier.Gonzalez.Huerta@bth.se (J. Gonzalez-Huerta), Michael.Dorner@bth.se (M. Dorner), aivars.sablis@saftehnika.com (A. Sablis).

https://doi.org/10.1016/j.jss.2023.111649

are happier, more innovative, more engaged, more motivated, more productive, more accurate and able to solve more complex tasks than manager-led teams (Janz et al., 1997a; Cohen and Bailey, 1997; Langfred, 2007; Bernstein et al., 2016). Autonomy improves the quality of work life (Kalliamvakou, 2017); purpose-driven, responsible individuals do not want to be managed or controlled (Rey et al., 2019). Yet, enabling the autonomous work of multiple teams working in parallel (autonomy at scale) is a challenge due to coordination complexity (Bernstein et al., 2016) and having too many dependencies between teams (Moe et al., 2021), and is sometimes referred to as a matter of managerial art and science (Mankins and Garton, 2017). The notion of "the boundaries of needed coordination and alignment" (Rey et al., 2019) are said to be those absolutely necessary. However, constrained autonomy is not well researched. Besides, self-centered autonomy can suffer from the association with ambiguity, inefficiencies, organizational chaos (Mankins and Garton, 2017) or simply opportunistic behavior. Therefore, some researchers associate autonomy with the freedom to make decisions and the responsibility over the completion of work (Janz et al., 1997b). Yet, the practical advice for how to distribute decision-making in a large organization and enable autonomy at scale are still missing.

In this article, we describe organizational decentralization at Spotify, the new generation agile company with iconic ways of working that promote team autonomy and employee empowerment and attempt to understand the mechanisms that enable team autonomy at scale.

2. Background: Spotify squads

Spotify, the world-leading provider of music streaming services, evolved its own ways of working, which became iconic for many new-generation agile organizations. The culture of engagement and teamwork is emphasized in the organizational structures of Spotify and the unique terminology used to describe them (squads, tribes, missions). Squads are engineering teams at Spotify, which were initially set up to feel like mini start-ups, be highly cross-functional and self-organized (Smite et al., 2019) and be able to prototype, test, code, deploy, operate, and A/B test features and services independently of each other. A squad would typically employ a mixture of backend engineers, web engineers, and data scientists to ensure the ability to develop and deliver end-to-end functionality. However, there are also specialized squads with only backend engineers or only web engineers. Spotify is a purpose-driven organization (Rey et al., 2019) with missions (business areas) as organizational units that unite several tribes (departments) working in a particular functional area. Missions are formed to be independent and, if possible, co-located, to ease the coordination and decision-making.

In the spring of 2020, Spotify employed ∼500 squads in six missions (business areas) divided into ∼50 tribes (departments) developing and maintaining thousands of services, mobile features and data pipelines. As the company and its product portfolio grew in size and complexity, maintaining autonomy became increasingly difficult. Some organizational structures have been abandoned (such as chapters Mankins and Garton, 2017), some have decreased in popularity (such as guilds Smite et al., 2019; Šmite et al., 2020), and some new structures have been introduced (such as missions). Yet, no organizational restructuring alone can alleviate the need for cross-squad collaboration and coordination. In the following two examples, we illustrate the scale of cross-squad coordination. Fig. 1 shows the number of technical dependencies in a network of backend microservices. A squad working with the backend may own one or more microservices. Few are isolated (see the periphery of the graph).

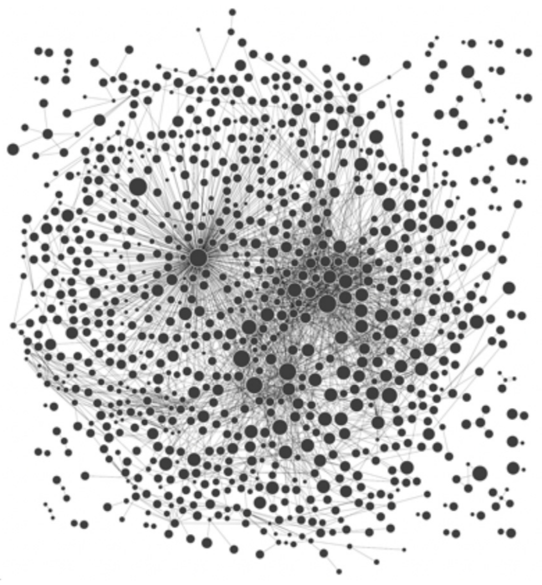


Fig. 1. Network of backend microservice dependencies. Every node is a service, every edge is a direct call obtained from the service catalog and service discovery data.

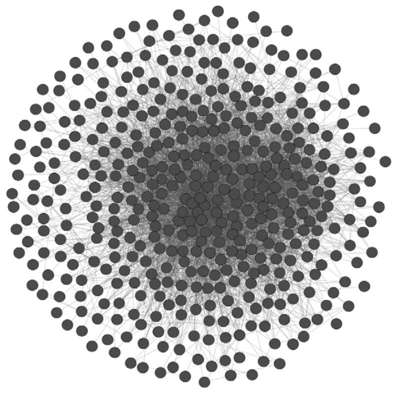


Fig. 2. Task dependencies between squads. Every node is a squad, every edge represents one or multiple GitHub pull requests connecting the squad authoring a pull request with a squad that owns the repository.

The majority are interrelated, often with services owned by other squads (see the center of the graph). Cross-squad work coordination may be resolved by requesting the owner squad to make the changes in their microservices (or other type of code), or by suggesting the changes in a pull request. Fig. 2 shows the network of such cross-squad pull requests accounting 435 author-reviewer combinations. In one year (2019), each squad, on average (mean) interacted with 26 other squads, which shows that squads perform highly interdependent work and contribute to many repositories owned by others.

To know how Spotify enables squad autonomy on a large and continuously increasing scale and what challenges they face, we looked at how work execution is organized. We asked six squads responsible for deploying features or services about their perception of autonomy, executed authority and its constraints, mechanisms enabling the work execution and barriers to their autonomy.

3. Details of the research study

To understand team autonomy at scale, we conducted an exploratory single-case study (Runeson et al., 2012) at Spotify,

Table 1
Profile of the studied squads.

Squads	Location	Short description	Members			Key reflections on own autonomy (based on direct squad member quotes from the retrospectives)
			Total	New hires	Consultants	
Alpha	Sweden	Well-performing	7	2	2	*"…We are a part of designing our future – what and how we build. It's not someone else, WE are defining it".*
Beta	Sweden	Challenged, recently redesigned	7	2	3	*"There is a lack of skills, confidence or competence, also resources, to do something"; "We have a this-has-always-worked syndrome"; "There is no one single person who owns the process and drives it, but it has become better in the last two months"; "I agree. Don't you think it has to do with [Squad Manager] who joined us?"*
Gamma	Sweden	Well-performing, somewhat unstable	6	3	–	*"Vision and mission for the squad is defined by ourselves. We sit together and decided what we want to do. People want us to own more stuff, but it' up to us to keep the scope".*
Delta	Sweden	Challenged, recently redesigned	8	2	–	*"Difficult for a team to be autonomous, a lot of politics go around – manager often helps with that"; "We can't do everything ourselves. Architecture does not support that. Autonomy ends at the boundary of our turf".*
Epsilon	USA	Well-performing, recently established	8	3	–	*"We can only be as good as are the parts we depend on. [We need to] keep track of the health of dependencies".*
Zeta	USA	Newly established	5	1	–	*"To truly be autonomous we need to remove all dependencies, which is an idealistic idea. If we were a small company, we would be able to have control of the tech stack, do more work, own more work, make decisions about our work ourselves. Dependencies hinder the ability to control our own destiny".*

based on qualitative and quantitative data from various sources, including:

- Squad retrospectives conducted with squad members,
- Interviews with squad managers,
- Social network analysis survey,
- Archival analysis,
- Process and organizational descriptions from the internal blog.

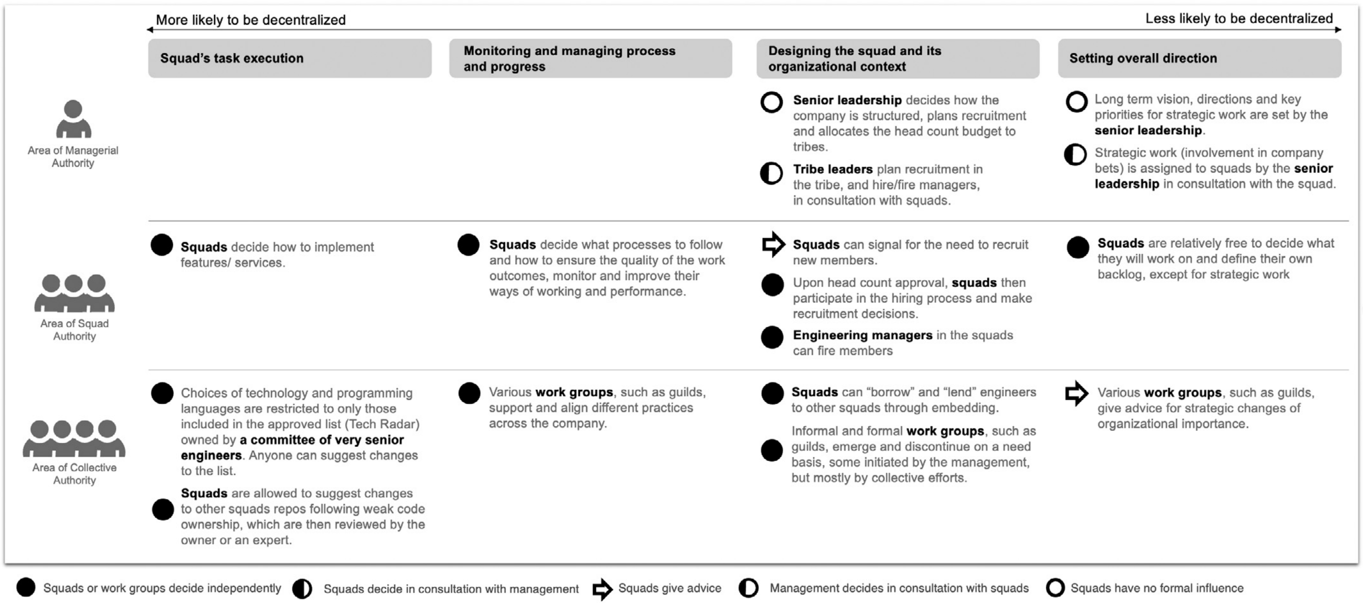
The objective of our research was to understand the degree of squad autonomy in a large-scale environment, barriers to squad autonomy, and mechanisms that help enable autonomy at scale.

Analysis of the perception of squad autonomy, executed authority and its constraints, and barriers to squad autonomy is based on the data gathered through interviews and retrospectives. In May, October and December 2019, we ran retrospectives with six squads (here called Alpha, Beta, Gamma, Delta, Epsilon and Zeta, see Table 1) and interviewed their engineering managers (managers of five out of six squads). Squads were selected by convenience sampling with a prerequisite to have challenged or recently formed squads (Alpha, Gamma and Zeta) and well-performing squads (Beta, Delta and Epsilon). Squads were recruited through the call for volunteers.

Retrospectives followed a structured 2,5-hours long agenda. We asked participants to individually report and later discuss enablers and hindrances to the squads' autonomy in a DAKI format (Drop-Add-Keep-Improve) (Caroli and Caetano, 2020) and discussed the squads' dependencies (coordination with experts and other squads, meeting fora for work coordination). Researchers took detailed close-to-transcript notes (later verified with the participants) and recorded the drawings. Interviews with squad managers focused on gathering general information about the squads and the work they are assigned to do, as well as discussing squad performance and factors affecting the squads' ability to execute their work. Detailed notes were taken during these interviews for further analysis.

The **qualitative analysis** started by identifying the level of squad authority and the distribution of managerial authority. Two first authors performed thematic coding of the detailed notes from team retrospectives and interviews with the four core themes based on the levels of authority suggested by Hackman (1986) and the decision-making authority being squads and managers (Hackman, 1986; Lee and Edmondson, 2017). For example, the following squads' member statements were assigned three codes [Squad-made decision], [Squad's task execution] and [How to implement features/ services] – *"I like that we can control and decide how we plan our sprints. We have a lot of control"* and *"What and how we build – it's not someone else, we are defining it ourselves"*. During the analysis, however, we decided to extend the primary themes that were based on the binary view of authority decentralization suggested by Hackman (1986) and followed by Lee and Edmondson (2017). Instead of squad-made vs management-made decisions, we also coded decisions made by the management in consultation with the squads, by the management based on squads' recommendations or input, by the squads in consultation with the management, and, finally, what is more important, we identify the area of collective responsibility, which is one key extension of the current literature. For example, the following quotations were coded as [Collective responsibility], [Squad's task execution], [Constrained choice of technology], and [Hindering factor]: *"A single authority that says what we should use – it's anti autonomy. We should be able to make such decisions inside the squad"*, and *"We need to standardize across squads. Standardization makes us more autonomous, because the policy applies to everyone – it is therefore both adds and hinders autonomy"*.

Next, codes from different squads were compared to determine whether levels of authority were perceived by all squads similarly or depended on some squad characteristics or the context. For example, a squad working on less prioritized tasks affected its ability to recruit members, in other words, to design its organizational context, as evidenced in the following quotation "*Hard to advertise this squad, hard to prioritize, so candidates came top down*" in contrast to another squad's experience: *"We are not in control of the headcounts. I am fine with that. But we are in control of the competences we need"*. The result of these steps of qualitative analysis was used to draft our understanding of authority distribution at Spotify, which was then combined with the process and organizational descriptions from the internal blog and verified with the representatives from the company management to derive the designed distribution of authority (Fig. 3).



More likely to be decentralized — Less likely to be decentralized

	Squad's task execution	Monitoring and managing process and progress	Designing the squad and its organizational context	Setting overall direction
Area of Managerial Authority			○ **Senior leadership** decides how the company is structured, plans recruitment and allocates the head count budget to tribes. ◑ **Tribe leaders** plan recruitment in the tribe, and hire/fire managers, in consultation with squads.	○ Long term vision, directions and key priorities for strategic work are set by the **senior leadership**. ◑ Strategic work (involvement in company bets) is assigned to squads by the **senior leadership** in consultation with the squad.
Area of Squad Authority	● **Squads** decide how to implement features/ services.	● **Squads** decide what processes to follow and how to ensure the quality of the work outcomes, monitor and improve their ways of working and performance.	⇨ **Squads** can signal for the need to recruit new members. ● Upon head count approval, **squads** then participate in the hiring process and make recruitment decisions. ● **Engineering managers** in the squads can fire members	● **Squads** are relatively free to decide what they will work on and define their own backlog, except for strategic work
Area of Collective Authority	● Choices of technology and programming languages are restricted to only those included in the approved list (Tech Radar) owned by **a committee of very senior engineers**. Anyone can suggest changes to the list. ● **Squads** are allowed to suggest changes to other squads repos following weak code ownership, which are then reviewed by the owner or an expert.	● Various **work groups**, such as guilds, support and align different practices across the company.	● **Squads** can "borrow" and "lend" engineers to other squads through embedding. ● Informal and formal **work groups**, such as guilds, emerge and discontinue on a need basis, some initiated by the management, but mostly by collective efforts.	⇨ Various **work groups**, such as guilds, give advice for strategic changes of organizational importance.

● Squads or work groups decide independently ◐ Squads decide in consultation with management ⇨ Squads give advice ◑ Management decides in consultation with squads ○ Squads have no formal influence

Fig. 3. Level of decentralization of authority by decision area.

Finally, we performed thematic coding of factors that hinder and enable squad autonomy based on squad retrospectives and identified barriers related to the squad characteristics and contextual factors that emerged in the comparative analysis of the levels of authority across squads. Barriers to autonomy are discussed in Section 5 and summarized in Fig. 5, while the enabling factors combined with the process and organizational descriptions were used to derive Spotify's practices to foster autonomy at scale, which are discussed in Section 6.

To capture squad interactions (see Fig. 6), we asked participants of the retrospectives to report important work connections, as many as they determined necessary, in a "free-recall" format. This was operated as a web-based survey distributed to all participants after the retrospective. Respondents recorded each work connection as a separate entry in the personal Google sheet, including the contact's name, the content of each connection with an ability to select knowledge received, knowledge transferred, work-related information received, work-related information transferred, administrative information received and/or administrative information transferred. Further, for each identified contact, the participants also provided an answer on the frequency of coordination with the selected contact using the 5-point Likert scale (from Rarely to Sometimes to Every day). Our survey partially replicated a questionnaire conducted by Manteli et al. (2014). We followed a "realist" approach and relied on perceived individual networks, which are believed to correspond to the actual boundaries of social groups (Lee and Edmondson, 2017), and used to identify social networks before (Šmite et al., 2017; Manteli et al., 2014). To construct the networks, we first completed the list of recalled contacts from all respondents and clarified these with company representatives to verify the names, roles in the company, and location. We visualize four squad networks, omitting two squads (Epsilon and Zeta) because of low response rate. In the four squads, 21 of the 28 squad members returned the survey (75% response rate, ranging from 60% for Alpha to 88% for Gamma). Social networks were then visualized using the Fruchterman–Reingold layout algorithm (Lee and Edmondson, 2017) based on the squad members, their contacts that they recalled and reported connections.

The **quantitative analysis** focused on illustrating our findings and demonstrating the internal validity or applicability of the findings across the company and beyond the squads included in the qualitative analysis. We start by illustrating the complexity of the work environment and cross-squad dependencies at Spotify in an image of backend services (Fig. 1) containing direct calls extracted from the service catalog and service discovery data, illustrating interdependencies and indirectly showing the level of work coupling on a company level. Next, we drew a network of task dependencies for squads based on the data extracted from the GitHub Enterprise (see Fig. 2). We used all 87,606 pull requests (PRs) from 2019, from which we excluded all squad-internally reviewed pull requests (44,36%), as we are interested in the cross-squad collaboration. In the network, we visualized unique author-reviewer combinations as the edges (total of 5659) and squads as the nodes (total of 435). This was done to show that the studied squads' coordination needs are not unique. The abovementioned data sources were provided by the company representatives. We have, therefore, not extracted them systematically or in connection to the studied squads. Next, using data from the employee database, we visualized the proportion of managers in the company to all employees in the engineering departments from 2018-01 to 2021-09 (see Fig. 4) to support the claim that managerial overhead in Spotify is relatively low despite the organizational growth. Finally, we drew another network illustrating mutual help and information exchange based on the data extracted from the Q&A system built as a corporate StackOverflow instance for the period between 2018-11-26 (the start of its use) and 2019-12-11, in which we connected question authors, respondents and commentators. In total, we have processed 2428 questions, 3112 responses and 3684 comments from 929 users (see Fig. 7). Here we show the company-wide use of one strategy we recommend as a mechanism supporting autonomy at scale.

The quantitative materials are used to understand the internal validity of our results and the representativeness of what we find in the studied squads in the context of the entire company. In other words, through the qualitative analysis we understand the level of autonomy that squads have and squad experiences with cross-squad alignment and collaboration, and through the quantitative analysis we understand how representative these needs for alignment and collaboration and mechanisms that support scaled autonomy are.

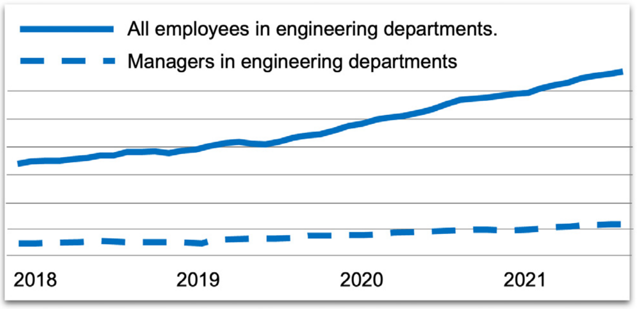


Fig. 4. Proportion of employees with the managerial role in the engineers departments.

4. Results – Decentralization of authority

We start by describing the level of autonomy and managerial authority based on the squad testimonies, followed by an outline of the planned distribution of managerial authority that combines the squad insights with the managers' testimonies and process and organizational descriptions (Fig. 3).

4.1. What do Spotify managers decide?

Autonomy and self-management are paramount for Spotify and are associated with empowerment, one of the core values that is believed to be the foundation of fast-paced product development and innovation and signs of humanistic, participatory management (Lee and Edmondson, 2017; Rey et al., 2019). Squads are given a great deal of authority to take squad-related decisions, which are only in certain circumstances and areas constrained by the management (see the level of decentralization of authority at Spotify by design in Fig. 3). The high level of decentralization can also be evidenced in the low proportion of the employees with managerial roles in the engineering departments, which remains rather stable despite the company growth (see data for 2018–2021 in Fig. 4).

Designing the organizational context high-level decisions related to organizational design and redesign at Spotify are largely centralized. Senior management decides how the organization should be structured and plans organizational growth. In its turn, tribe management plans and leads the recruitment efforts, especially when hiring (and dismissing) managers.

Setting overall direction: Strategic decisions regarding the long-term vision, directions and key priorities at Spotify are set by the senior leadership. Further, all Spotify missions and tribes perform quarterly planning — these decisions are taken by the tribe leadership in consultation with the individual squads. In addition, the senior leadership reviews new initiatives quarterly and sets priorities for strategic work, called company bets. For example, when Spotify entered the Japanese market, it required significant changes to the core software functions such as the search functions and metadata fields, as well as demanding new functions such as having karaoke mode. Company strategic initiatives usually involve squads from different tribes and even missions, and thus require resource allocation outside of the scope of the squads- or even tribe-focused planning. Such planning usually starts by approaching squads and including them in design discussions and working groups.

4.2. What can Spotify squads decide?

When asked to reflect on the squad autonomy, everyone agreed that squads, in principle, are granted significant authority, even though not all squads felt equally autonomous. The well-performing squads perceived exploiting higher levels of autonomy, while the challenged squads were said to be less autonomous leading to increased management involvement. An engineer from Gamma explained that to be independent squads ought to prove their accountability: *"As long as you can show that you are creating value for the company, the company will believe in you. People around you will know that what you are doing [...] is something useful. So, they will let you keep doing it"*. Besides, squad autonomy also depended on the position of the squad, and its work in the organization, i.e., how much the squad is dependent on others and how much others depend on the squad in terms of technical dependencies and competencies. In the following, we describe what type of decisions Spotify squads can take independently, and where their autonomy needs to be constrained.

Executing squad's tasks Decisions regarding work execution at Spotify are fully decentralized and management does not directly interfere with squad-internal decisions on how to implement a feature or a service. To a large extent, squads independently decide how to code a feature or service, with few exceptions such as where squad autonomy is sacrificed because squads do not work in isolation, and some decisions must be taken collectively. The key reasons to constrain squad independence are twofold – necessity to align on the choices of technology and programming languages and the necessity to coordinate task dependencies.

Autonomous squads should, in theory, be able to choose the technology best suited for solving their tasks. However, in practice, there is a need to align the use of programming languages, technologies, frameworks, processes, and infrastructure to avoid maintainability problems and increase the ability to collaborate and help each other (read about TechRadar, Spotify's collective effort to align technical decisions in the next section).

Squads' freedom is also constrained by the technical dependencies with other squads, which we have found surprisingly many, despite the company's attempts to decouple the architecture by design (microservice architecture principles) and by action (code ownership principles and continuous re-engineering). The number of cross-squad and even cross-tribe pull requests especially in mobile client development (see Figs. 2 and 3) is huge, and these technical dependencies have been acknowledged as the major challenge by all studied squads. As an engineer from Delta stated: *"We can't do everything ourselves; architecture does not support that. Autonomy ends at the boundary of our turf"*, while someone from Zeta noted: "*Dependencies hinder the ability to control our own destiny"*. Changes in others' code are unavoidable, especially on the client-side where there is no good equivalent to the backend microservice architecture. We learned that some squads are significantly affected by these uncontrolled dependencies, as one engineer from Beta explained: "*We write directly to others' databases, and others write to ours. Interfaces to other teams are unspecified. We are stuck in a way, which prevents us from being independent*" (read about the ways to coordinate technical dependencies in the next section).

Monitoring and managing process and progress: Decisions related to the process management at Spotify are largely decentralized and decisions on how to perform the actual work in a particular squad are driven by the squad. Planning decisions, progress monitoring and quality assurance tasks are also all responsibilities of the individual squads. In our study, we found that well-performing squads at Spotify actively followed their progress by gathering and analyzing squad-related performance data, which helped to continuously improve their internal processes. As a member of Alpha explained, *"The [performance] data is our reference point. We know that work with a lot of dependencies takes 2,5 times longer, so what can we do about that? It gives us a lot more freedom – Let's try this thing; No, it did not really help, so let's try something different"*. Although not all squads have such advanced progress data monitoring habits, all squads mentioned having retrospectives and experimenting with ways of working, except for one challenged squad, Beta, that felt they did not succeed with improvements, as the squad members described *"being focused on something too much, and not seeing problems around"*, *"not questioning things"*, and having the so-called *"this-has-always-worked syndrome"*. Admittedly, failing to manage own processes might considerably hinder teams ability to self-manage (Hackman, 1986). As explained by the members from Delta — the squad started out with a clear scope but over time received more diverse tasks into their backlog; to optimize productivity, each engineer became responsible for their own tasks, which decreased teamwork. As a member of Delta explained: "*Because we are doing so many different things at the same time, it is impossible to replace each other*". Eventually, in the absence of reflection and corrective action, the squad had too many diverse tasks with urgent priority. Because members focused on their own tasks and thus individual goals, joint goals became blurry, and the squad performance suffered.

Designing the organizational context: Many squads have substantial authority over the recruitment of new squad members. Any squad can signal the need for more people, which is considered by tribe leadership. The following recruitment process is typically led by the squad together with their engineering manager, who also has the authority to remove squad members if seen necessary. However, not all squads are equally successful in getting new members. Members of Delta explained that some squads, like theirs, demand very specific competences, which are hard to advertise. There might also be differences in how prioritized and visible the work done in some squads is, which affects member recruitment.

Setting overall direction: Squad-related strategic decisions include choosing the squad's scope of work, prioritizing work in the backlog, and taking decisions regarding the future of the features and services that a squad owns, which might require taking decisions with economic consequences (Guzzo and Dickson, 1996). The key sources of work tasks for squads include market feedback and product development ideas regarding their area of responsibility. For example, there is a squad in Spotify, which owns "Spotify experience in the car" and together with their product manager plans the features and services to be developed or improved. Squads do not independently determine their own direction but certainly have a big influence on their goals and scope. Work planning for the squad is performed in sprint planning meetings and is the sole responsibility of the squad, except for the cases when squads are assigned to strategic initiatives, i.e., company bets. Among the squads we studied, some reported having a very clear scope with full control over it, as one explained: *"We sit together and decide what we want to do"*. Yet, some squads admitted that their goals are *"too fluffy"*, which is the consequence of not managing their processes well. The studied squads explained that internal consensus and clarity in what the squad owns and commits doing is important to ensure their accountability. Finally, although squads do not decide which projects or product areas they work on, they have relatively large freedom to contribute and innovate within the business area they belong to. However, we learned that high performing squads desire to contribute more broadly and be even more autonomous when it comes to cross-company contributions. As someone from squad Alpha explained, *"We are defined by the mission of what we do and that sometimes limits us to do more. We could innovate in different places and do more, but we are limited"*.

4.3. Where are Spotify squads obliged to compromise, align or coordinate?

Autonomy at Spotify is clearly not understood as anarchy. The studied squads acknowledge that the scaled autonomy starts with the end-to-end responsibility for their work, which requires further support in the form of formal rules guiding collective action. Collective efforts concern the formation of various work-groups and squad-to-squad interactions that occur naturally, on a need basis. In what follows, we describe several formalized work groups and efforts, as well as principles and actions that Spotify engineers follow to self-organize.

Executing squad's tasks To support the alignment of the choice of technologies used by the squads, Spotify established a central committee of very senior engineers who own the list of accepted technologies called TechRadar, and a well-defined process for proposing technologies and trying them out. Anyone can and is encouraged to suggest new technologies that might be of interest for the company. In our study, we found that not everyone agrees with the restrictions that TechRadar implies. Some members we talked to strongly object to the limitations of professional freedom, calling this practice an "anti-autonomy". Yet, many understand the importance of enabling constraints, e.g., limiting the use of unpopular or unsupported technologies. In a way, limitations to someone's autonomy might enable others' autonomy. They help to be more resilient and avoid being stuck with the code written in a language that nobody knows, when the original developers leave or move, and to improve evolvability of the products, allowing squads to help each other and contribute to the majority areas of the codebase.

Work coordination across squads and joint decision-making is also the consequence of selected code ownership principles followed at Spotify — being allowed to change code owned by other squads (weak code ownership principles, i.e., taking responsibility for your code, letting others change it, and keeping an eye on those changes Fowler, 2006). When the squad's work impacts other squads, engineers are expected to consult the others either informally or through formalized action. Large changes are documented in the form of so-called Requests-For-Comment (RFC) which are sent to a wider audience for feedback, to ensure that there is an agreement about the emerging changes and dependencies. Smaller changes are handled through the GitHub pull requests (PRs). The pull requests are typically reviewed by the owning squad, which is responsible for coordinating the changes and keeping an eye on the technical health of their code repositories. In some cases, changes are reviewed by members of a third squad, not authoring the change or owning the repositories, if changes are critical or interdisciplinary.

Monitoring and managing process and progress: Spotify culture and ways of working are embedded in the organizational design. Our previous research at Spotify illuminates the workgroups called guilds that collectively monitor domain- or technology-related practices and discuss the needed improvements (Smite et al., 2019; Šmite et al., 2020). For example, processes of quality assurance are discussed in the quality guild, while web guild and C++ guild are concerned with the questions related to practices of programming in a particular language or technology, many of which also maintain active support channels to help those with questions. A template for squad retrospectives developed by the agile guild is an example of a concrete outcome of collective efforts. Beside the guilds, there are other collective efforts that emerge on a need basis, such as "Friday-feels" circles that bring together managers, or book clubs where interested participants discuss insights and applicability of ideas from literature. The outcomes of these groups often provide recommendations for improving various work processes. For example, the No-meeting-Wednesday initiative, which emerged in response to numerous complaints and discussions during the COVID-19 pandemic time of working from home. New processes and practices are usually piloted by the dedicated workgroups and recommended for a broader use if found valuable. The results of pilot studies are spread in the company through various means, internal blog posts and presentations at various group meetings being the most common ones. Whether the suggested new practices will be followed by the squads largely depends on the level of social integration, curiosity, and awareness of the individual squad members of the collective initiatives.

Designing the organizational context: As a typical post-bureaucratic organization (Burns and Stalker, 1961), Spotify promotes network structures of control, authority, and communication, as evidenced in the presence of collective control mechanisms. Various workgroups and groups of interest, such as communities of practice called guilds (Smite et al., 2019) or the management support group called "Friday feels" emerge and self-manage primarily by collective efforts. When groups are formally recognized, they often are mandated to make certain decisions or recommendations to the leadership.

When it comes to solving squad-centered experience or expertise gaps, it is not uncommon to mutually agree on a solution with another squad, i.e., borrowing and lending members (the practice called "embedding", usually associated with member mobility for three months). These decisions are often initiated by individual engineers who want to increase their competence or squads who need resources approved by engineering managers.

Setting overall direction: Although Spotify workgroups described in our work are not mandated to make strategic decisions or setting overall direction, they are encouraged to voice their opinion and bring to light the operational evidence as input to strategic work. Examples of collective efforts that reached strategic level include the inclusion of Rust, an emerging programming language, into the TechRadar as a result of curiosity-based experiments, and subsequent hands-on experiments (PoCs) in the C++ guild, and a bottoms-up initiative to set-up support for build systems for web components that led to the creation of a dedicated team that rolled this initiative out for all of Spotify.

5. Discussion

In this paper, we presented the perceptions of squad authority and the levels of organizational decentralization by design. In this section, we discuss first discuss the factors that hinder squad autonomy based on the analysis of the squad testimonies, their characteristics and contextual factors that affected the ability of the squads to exploit the designed levels of authority. We then present the strategies for enabling autonomy at scale based on the various forms of organizational authority decentralization and support. We use the results of quantitative analysis of the squad self-reported social networks and information sharing networks from internal systems to further illustrate and support our recommendations.

5.1. Barriers to Squad autonomy

Here, we present the factors that were pointed out as barriers to squad autonomy, which we mapped to the levels of authority (Hackman, 1986) (see a summary in Fig. 5).

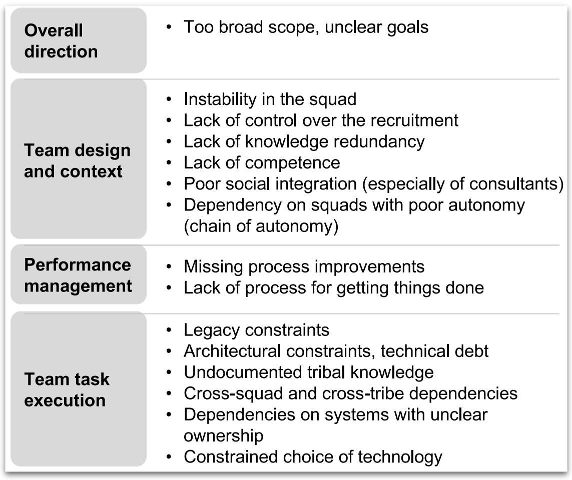
Our results demonstrate that despite the efforts to decentralize the organization by design, squads' autonomy might be limited by factors beyond managerial control. For example, the squad immaturity (lack of knowledge redundancy, lack of competence) negatively impacts autonomy. This is consonant with the related work that associates autonomy not only with the freedom to make decisions, but also with the responsibility over the completion of work (Janz et al., 1997b) and work that links team independence with the self-sufficiency in skills (Kalliamvakou, 2017). Autonomy and responsibility are also found to be a source of stress (Grant and Parker, 2009). In our study, we found autonomy to cause pressure related to the necessity to coordinate the work and manage mutual dependencies.

Another large group of barriers to squad autonomy is rooted in injured teamwork and failure to self-manage (too broad scope, unclear goals, missing process improvements, lack of process for getting things done, poor onboarding).

Some context factors (lack of control over recruitment, team instability) can also negatively affect squad autonomy.

While previous factors were related to the squads internal doing, another group of factors that hindered autonomy at scale relates to technical, human and task dependencies. Barriers here contain architectural and legacy constraints, technical debt, the need to compromise with old systems and clients, and undocumented tribal knowledge. Evidently, despite the architectural decoupling (microservices) which has been previously found to enable independence (Kalliamvakou, 2017; Jamshidi et al., 2018), we found that the implementation of code ownership also matters. We learned that weak ownership principles lead to squads depending on others and the success of cross-squad collaboration depends on the social integration of individual squad members in the organizational network, which has been also pointed out in other studies related to large-scale software development (Šmite et al., 2017). Finally, work on interactive parts of the system used by others (referred to as increased visibility) means that squads receive many questions and requests for code review, limiting their autonomy. Interestingly, while mandatory code reviews in related studies are found to be related to increased coordination, they are also related to aligning the team internal practices with those of other teams' (Kalliamvakou, 2017).

Finally, we learned that the lack of squad autonomy does not only affect the squad concerned; when working at scale it also affects other squads as a chain reaction, which we refer to as the *chain of autonomy*. This is especially evident when squads are hindered by task dependencies with many incoming requests for code reviews or code changes. When coupled with the lack of experience or competence, squads are likely to be congested (when arrival of tasks exceeds the ability to close them), and this is found to also cause the congestion in the whole network (Cantor et al., 2016). The concepts of the chain of autonomy and network congestion indirectly substantiate Bungay's argument that increased alignment increases the autonomy an organization can grant (Bungay, 2011). Our research further shows that when teams' capabilities or goals are not aligned (as often the case in a growing organization), it is hard to rely on the designed level of autonomy.



Overall direction	• Too broad scope, unclear goals
Team design and context	• Instability in the squad • Lack of control over the recruitment • Lack of knowledge redundancy • Lack of competence • Poor social integration (especially of consultants) • Dependency on squads with poor autonomy (chain of autonomy)
Performance management	• Missing process improvements • Lack of process for getting things done
Team task execution	• Legacy constraints • Architectural constraints, technical debt • Undocumented tribal knowledge • Cross-squad and cross-tribe dependencies • Dependencies on systems with unclear ownership • Constrained choice of technology

Fig. 5. Barriers to Squad Autonomy.

5.2. Strategies for autonomy at scale

In this section, we outline the mechanisms that help Spotify address many of the aforementioned challenges and to orchestrate the interdependent work of hundreds of autonomous squads while keeping organizational control to a minimum.

5.2.1. Strategies for sharing the codebase

Like in related studies that suggest that large-scale collaboration requires practices that are more typical for open-source software projects such as reduced communication, independent work, and self-organization enabled through GitHub (Ingvaldsen and Rolfsen, 2012), we also found code maintenance in GitHub as an important coordination mechanism. In the following, we present two practices that were found essential to enable effective collaboration on the codebase with increased autonomy.

Formalized code ownership To ensure that squads can have end-to-end responsibility for the developed functionality and minimize the handovers, Spotify engineers are allowed to change code owned by other squads, following *weak code ownership* principles (Fowler, 2006) (as opposed to *collective code ownership* which is equivalent to no ownership or *strong code ownership*, which often challenges the end-to-end responsibility allocation on a functional level). We found that formalizing code ownership in general is a prerequisite for the well-functioning of the cross-squad collaboration at scale. When a piece of code has an owner, there is a clear point of contact for those with questions and a clear quality guard when changes emerge. The absence of code owners hinders squads, as a member of Gamma explained, "*Some systems are not owned by anyone. If you need something from them it's impossible to get any help*". Code changes are typically reviewed by the owning squad, which is responsible for coordinating the changes and keeping an eye on the technical health of their code repositories. In some rare cases changes are reviewed by members of a third squad, not authoring the change or owning the repositories, if changes are critical or interdisciplinary.

Self-managing code ownership: Unfortunately, it is not uncommon that weak code ownership leads to slow pull request handling and bottlenecks and slows the squads' progress down. As a member of Alpha explained: *"We are waiting for others to review our stuff. In the team we are quite quick to review [internal PRs] because we can talk physically, but other teams can be quite slow because they have their focus. You have to poke them a lot sometimes"*; and another member admitted *"If someone puts a PR in our stack, it takes 4–5 days before they are done. [...] It's too slow and we are delaying it because... I don't know why"*. Many agree that this could be addressed by establishing better "contracts" between squads specifying mutual expectations and responsibilities, or just better processes for getting the pull requests done (e.g., dedicating time every morning to unblocking others). To optimize the dependencies, squads may also self-manage the code ownership. For example, Beta and Gamma negotiated and changed the ownership of the code repositories they depended on to increase squad autonomy. As a member from Gamma explained, "*We took over [Beta's] work because they were not able to help us to do that. We now own a part of their problem. We are more autonomous, though we also have more work*".

5.2.2. Strategies for achieving alignment

Autonomy and self-management are found to predispose companies to reduced coordination across teams (Ingvaldsen and Rolfsen, 2012). This is why, large-scale development efforts require alignment towards the clear vision or goal for a team and awareness of how the team's work fits in it (Kalliamvakou, 2017). Traditionally, this is achieved by enforcing more control and providing detailed instructions, which is proven to be a failing strategy in rapidly changing contexts. Instead, autonomy is increased by increasing the alignment through limiting the direction and supporting mutual adjustment (Bungay, 2011). Management efforts that support alignment, called "directed opportunism", such as guidelines towards commonality, are also found to be the key to success in scaling agility in Atlassian (Kalliamvakou, 2017). Similarly, many other companies use Objectives and Key Results (OKRs), a framework for large-scale agile environments that attempts to involve employees in setting common goals across the organization. OKRs have been found to aid knowledge sharing and improved transparency between teams, however, the practice takes years to implement and is not always well received (Stray et al., 2002). In the following, we describe the practices that support alignment at Spotify.

Collective review of important changes: One inherent problem with scale is that it is nearly impossible to continuously follow what is being changed where. Spotify's answer to this is collective review of important changes. When the squad's work impacts other squads, engineers are expected to consult the others either informally or through so-called Requests-For-Comment (RFC) to ensure that there is an agreement about the emerging changes and dependencies. The RFC document is typically prepared and shared with the rest of the company or the concerned tribe, open for comments and improvement suggestions for a given period. Similar effects in Atlassian are achieved by organizing regular demos of teams' work in progress, which admittedly increase the needs for coordination to arrange events (Kalliamvakou, 2017).

Formation of parallel structures mandated to make decisions: As a typical post-bureaucratic organization (Burns and Stalker, 1961), Spotify promotes network structures of control, authority, and communication, as essential mechanisms for collective action. These are, for example, numerous workgroups, such as the Technology Architecture Group that owns TechRadar and guilds, which emerge on the need basis by self-organization or as a response to a managerial directive and involve members from different squads, tribes, and sometimes even locations. The formally recognized groups are mandated to make decisions and might have a budget. Similar groups for making collective product-related decisions called communities of practice have also been found in Ericsson (Paasivaara and Lassenius, 2014; Moe et al., 2021).

Constrained choice of technologies: Implementing changes in other's repositories can be troublesome if they are written in a unique or rare programming language or using old unsupported frameworks, as it might limit squads' ability to implement changes independently. Clearly, with scale the number of used technologies increases, while the awareness of what is implemented using which technology decreases. To eliminate such situations, squads discuss what technologies shall be accepted, the list of which (called TechRadar) is maintained by the Technology Architecture Group. The different tools, apps, and infrastructures recommended for each engineering task are available in the developers' Marketplace called *Backstage*.[1]

Alignment of practices through tutorials and templates (Golden Paths): The wide variety of technologies and tools combined with the large number of developers and development locations makes it not only difficult to get onboarded as a new hire, but also for experienced engineers to produce consistent outcomes, especially when contributing to tasks outside of one's comfort zone. To tackle these challenges, Spotify developed so-called *Golden Paths* – tutorials to introduce engineers to technologies or best practices and templates that guide them on how to create a concrete type of services. Golden Paths are similar to coding guidelines and tutorials used in many other organizations (Kalliamvakou, 2017). The development of Golden Paths is organized in the different workgroups, such as guilds dedicated to a particular technology (Web and Backend), while the resulting documentation is maintained by a dedicated tribe.

5.2.3. Strategies for networking

While some argue that achieving agility in large-scale projects requires blending in plan-driven approaches and increasing coordination and management efforts (Dingsøyr et al., 2018; Petersen and Wohlin, 2010), some studies show that planned coordination through forums like Scrum-of-Scrums is inefficient (Paasivaara et al., 2012). This has been one reason for an increased interest in mutual adjustment as the core coordination mechanisms that enables alignment of teams' efforts and heavily relies on the social networks of individuals and teams (Šmite et al., 2017; Moe et al., 2021). In the following, we describe the networking practices that support autonomy at Spotify at scale, which we also illustrate with the social networks of the four studied squads (see Fig. 6).

Horizontal communication patterns: Spotify squads are highly collaborative and social, reaching out to other squads, experts, workgroup members, etc. This is evidenced in the way task interdependencies are handled (Fig. 2), and the way information is shared (Fig. 6). In fact, an individual squad with experienced members might have tens of contacts outside the squad maintained on a regular basis (see examples of external networks of Beta and Delta in Fig. 6).

The spontaneous interactions of the engineers vary. They include acquiring the knowledge missing internally in the squad, providing support to others, and dealing with task interdependencies. The communication typically bypasses the managerial hierarchy, also known as "going role to role" in Zappos and other participative organizations (Bernstein et al., 2016). However, the ability to communicate horizontally highly depends on the individual squad member's social integration and contact network, in the absence of which the members might turn to their manager for help. However, in principle, mature and newly established squads as well as senior and junior employees at Spotify, as in Zappos (Lee and Edmondson, 2017), enjoy autonomy.

Social integration: The studied squads discussed the importance of maintaining a good contact network and knowing who knows what, as being well integrated into the company's network helps to coordinate the work efficiently. A member of Delta explained that getting their PRs done requires a buy-in and priority from others. These depended on the negotiator's social connections, and the ability to influence others. As someone from a relatively new squad, Zeta, explained: "*You have to use your network or find people who have been longer in Spotify, [they] have a larger network. [...] [Otherwise], sometimes we need to post to a squad's Slack channel: "Hey channel, can anybody help me out here?"*." Squads that perceive being successful in negotiations often have pioneer engineers who know nearly everyone (like Beta and Delta in Fig. 6). Being aware of the organizational network, who knows what and who does what is important not only when working

[1] Open-sourced tool released by Spotify https://backstage.io/.

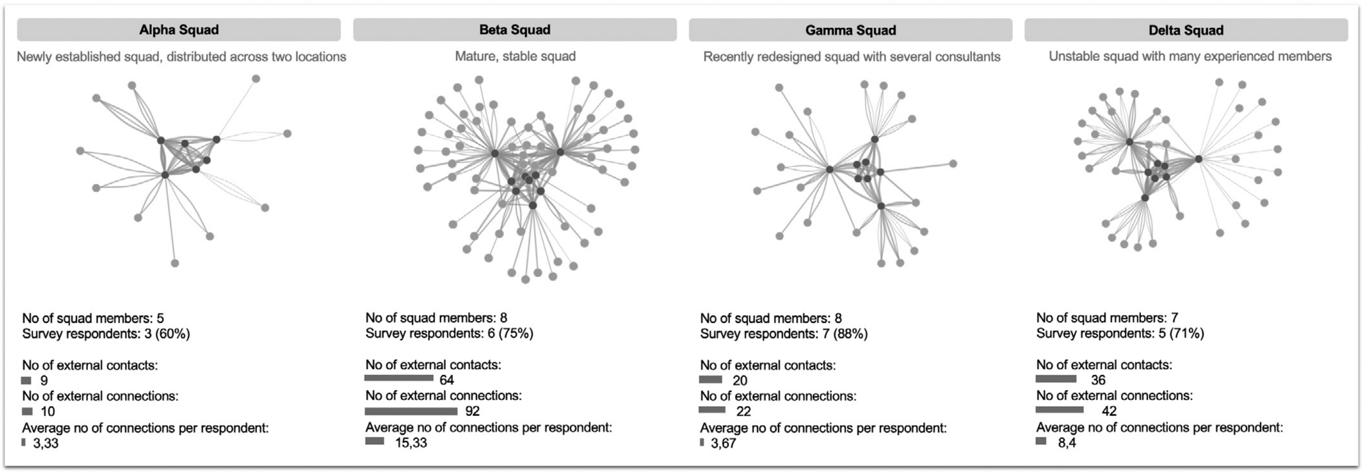

Alpha Squad
Newly established squad, distributed across two locations
No of squad members: 5
Survey respondents: 3 (60%)
No of external contacts: 9
No of external connections: 10
Average no of connections per respondent: 3,33
Beta Squad
Mature, stable squad
No of squad members: 8
Survey respondents: 6 (75%)
No of external contacts: 64
No of external connections: 92
Average no of connections per respondent: 15,33
Gamma Squad
Recently redesigned squad with several consultants
No of squad members: 8
Survey respondents: 7 (88%)
No of external contacts: 20
No of external connections: 22
Average no of connections per respondent: 3,67
Delta Squad
Unstable squad with many experienced members
No of squad members: 7
Survey respondents: 5 (71%)
No of external contacts: 36
No of external connections: 42
Average no of connections per respondent: 8,4

Fig. 6. Squad external networks of contact.

on interdependent existing products, but also when coordinating work on new products. As evidenced in the challenge faced by Zeta who are both new in the company and working on developing a new piece of software: *"Since we are building something new, it's unclear who we need to talk to"*.

Embedding: The practice of temporary mobility called embedding, the social integration with other squads for some time, is a practice encouraged at Spotify that helps grow social networks and foster inter-squad communication. Embedding has many positive effects, outweighing the pain of the temporary loss of a squad member. As someone from Delta explained, *"You can pair with an engineer who knows something you don't. This would help in doing pull requests and get an easier buy-in from another team. People could embed with us as well, to have people who could help us"*. Embedding has become popular also because it contributes to personal growth. A similar practice of moving around different teams to acquire first-hand knowledge of how others are working is also reported at Atlassian (Kalliamvakou, 2017).

Personal interaction: Despite Spotify being an international company with extensive experience with distributed work and excellent technological infrastructure, we learned that effective interaction still highly depends on the ability to communicate face-to-face. Co-location was mentioned frequently when discussing how squads negotiate with other squads. As one from Alpha noted, "*Other teams can be quite slow because they have their focus. You have to poke them a lot, sometimes through Slack, sometimes I can go and say – Hey!*" and another member added, *"It helps if you can explain what you are doing and why it is done like this"*. Such communication across offices becomes more challenging. As someone from Alpha explained, *"Sometimes we only see them in Slack, we don't really know them. If we knew them or met them in person, the way we would talk would be different. Problems would be solved much faster"*. Members who travel to other Spotify offices also acknowledged the positive effect of visits on cross-site communication.

Office spaces that facilitate socialization and networking: Spotify has invested a lot of resources and attention to provide engineers with numerous opportunities to socialize in the office. These include comfortable large dining areas and coffee corners, open spaces for seminars, video- and board-gaming spaces and even billiards and pubs. Being introduced to a friend of a friend or getting to know people who are interested in the same activities in such companies as Spotify is thus a common happening.

5.2.4. Strategies for information exchange

In contrast to teams working on isolated tasks or stand-alone components, teams working on creating a joint system in parallel with many other teams are accompanied by the feeling of being a piece of a system. As we discussed above, such systems are prone to congestion, which occurs when teams are tasked to deliver more than they can (Cantor et al., 2016). Evidently congestion is often caused by barriers to autonomy, the lack of competence or many task dependencies, which may negatively affect the whole company. One known congestion-avoidance mechanism for teams in plan-driven organizations is to ask management to limit their requests. But what happens in the contexts with increased level of autonomy? At Spotify, squads often help each other and increase awareness and visibility into their work that indirectly supports alignment (Kalliamvakou, 2017).

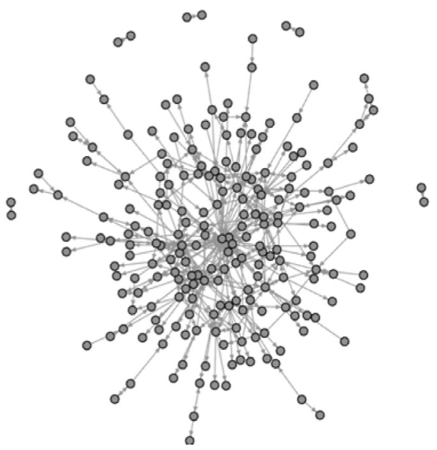


Fig. 7. Information sharing network. Every dot is an author of a StackOverflow entry tagged with #Backend, every edge connects questions with answers and comments.

Culture of mutual help: The culture of open collaboration and contribution beyond organizational boundaries is emphasized in Spotify's Band Manifesto[2] – *"we're all in this together"*. The success of the multi-squad development at scale highly depends on mutual help, which indirectly diminishes the autonomy (independence) of a particular squad. As an engineer stated, *"The number of dependencies is not a problem as long as no one becomes a bottleneck"*. One way for a squad to become a bottleneck for others and for the company is to constantly prioritize their own tasks from the backlog. In fact, previous research has linked individual autonomy with being less accessible for support (Drach-Zahavy, 2004), which evidently can be the case on a group level. This is challenging because squads depend on community help, on squads solving pull requests for others, and more experienced Spotifiers responding to queries and providing expert opinions about solutions to complex problems. Squads and individuals have a reputation of being responsive or not. This is related not only to availability for a conversation but also to personal activity in solving pull requests promptly and responding to queries received through open and focused information exchange channels such as Slack[3] and a corporate StackOverflow[4].

Tools for information exchange: Our analysis suggests that tools like Slack and StackOverflow are excellent facilitators of collaboration even in the absence of personal contact. For example, junior backend developers, who do not know who the experts are, often turn to the backend guild's Slack channel, which resembles a "support line" with hundreds of users and tens of members monitoring the channel, ready to provide answers (Smite et al., 2019). Similarly, the corporate StackOverflow accumulates answers to a wide variety of questions from technical questions to legal frameworks (e.g., GDPR) or even personal experiences (see an example in Fig. 7). In slightly more than a year of its existence 2428 questions, 3112 responses and 3684 comments were exchanged between the total of 929 users. Our analysis shows that half of the questions posted in StackOverflow are answered within thirty minutes and 59% of the accepted answers come from members of other tribes, which engineers with the question might not even know personally. Thus, those not knowing who

[2] The Band Manifesto, https://www.spotifyjobs.com/the-band-manifesto/.

[3] Slack at Spotify is one of the key communication tools used by individuals, squads and groups. It is also used to share knowledge, facilitate Q&A, and store an archive for everyone to use.

[4] Spotify's instance of StackOverflow. At Spotify this is a relatively new tool, in use since 2018, as a platform where users can post questions and seek advice, answer others' questions, comment and vote on posts.

knows what can reach out to basically everyone subscribed to a channel. More importantly, we found that 1/3 of the questions in StackOverflow are answered by the authors themselves. This is because engineers take the initiative to help others and post tips and tricks to ease the work of peers. Those who donate knowledge also receive points in the user reputation score in StackOverflow, which adds an element of gamification.

Demos: When squads build something good and potentially useful for others, it should be shared. Sharing at Spotify is done in many ways. Squads make demos, tribes circulate the latest developments in regular "Show & Tell" and "All Hands" sessions, there are a lot of "Lunch & Learn" sessions, where engineers and squads share interesting findings, many engineers at Spotify write blog posts about new tools and libraries they've developed, and, of course, post the news on Slack.

All these activities can be overwhelming and disturbing, as a member of a challenged squad Beta describes, *"...activities outside of development, going presenting at tribe meetings, contacting people – takes away the development focus and time. The expectation is that developers are not just developers, but ambassadors for the team"*. Yet, many squads acknowledge that maintaining their reputation is important as it creates the prerequisite for a higher degree of authority.

6. Conclusions

Many researchers and practitioners have been skeptical about the applicability of autonomous teams in large-scale and complex projects. This is one reason why large-scale agile projects often combine agile and traditional methods to avoid chaos (Dingsøyr et al., 2018; Bernstein et al., 2016), and in practice strive to simplify jobs by creating complex organizational structures expressed in increased managerial overhead (Petersen and Wohlin, 2010). However, uncertainty and frequent changes, that often accompany software development, make increased bureaucratization and control inefficient (Bungay, 2011). Decentralization and de-bureaucratization instead suggest that the focus shall be rather put on dealing with complex jobs within simple organizations (de Sitter et al., 1997). Our study is one example of a large decentralized organization developing software. Our results demonstrate that Spotify can empower its squads with a large degree of autonomy even at scale. Further, the company managed to keep a low number of employees with managerial roles despite organizational growth. What helps Spotify succeed? We found that the key is the ability of squads to self-organize – collaborate on the shared codebase, coordinate and align their efforts, share information and resources, and make informed decisions together. To facilitate squads with autonomy and ability to self-organize, the company promotes greater commitment and accountability, cultivates network structures and supports formation of collective fora that bypass organizational boundaries (i.e., typical mechanisms of a post-bureaucratic organization Burns and Stalker, 1961), while enabling just-necessary enabling constraints, which help balancing adaptability and reliability (Bernstein et al., 2016).

Of course, the Spotify journey is not problem-free. Our results demonstrate that despite the efforts to decentralize the organization by design, squads' autonomy might be constrained by factors beyond the managerial control. In our study, not all squads were equally autonomous or accountable, some squads helped others more and faster while some admitted being a bottleneck, some were better in informing about the changes while others prioritized their internal workload, some were better at handling and continuously removing technical interdependencies while others were too inexperienced to navigate themselves in the complex system evolution. Yet, as previously suggested (Elbanna, 2010), most squads can handle most of their mutual dependencies and interactions. In fact, no scaled agile practice, up-front planning, or supervision can yield similar results as also admitted by prior research (Bungay, 2011; Kalliamvakou, 2017; Paasivaara and Lassenius, 2014; Moe et al., 2021).

The company's main challenge is to continue ensuring team autonomy while growing. This is because being autonomous requires being socially integrated into the company, which takes time. As the Spotifiers from Alpha referred to a phenomenon called the chain of autonomy: *"If other teams are less autonomous, we, being a part of the same system, become less autonomous. If we want to do something and another team is blocked because they are not as autonomous as we are, there is a chain reaction"*. Because of these chain reactions, suboptimal solutions made by less mature squads working in isolation that result in the accumulation of technical debt might limit the future work of related squads. As another from Alpha observed: *"What affects us today is not our quality, but others' quality"*. This also means that maintaining autonomy for a continuously growing organization will be a constant challenge.

Several interesting directions for future research emerge from our findings. First, we recommend looking into the conditions of granted autonomy for immature and challenged teams and the ways of coaching teams to be autonomous. Second, it is reasonable to look at the team formation principles to maximize the ability of a team to be autonomous (e.g., having a healthy mixture of seniority at the company present in the team). Third, future research may investigate the relationships between organizational size and the very ability to employ autonomous teams, as well as the possible thresholds when autonomy becomes technically unmanageable.

Finally, one may wonder whether Spotify's case is unique or useful for others. The interest in autonomy and self-management emerge in many different contexts, not only within modern IT companies published in software engineering research, but also in other industries published as research on the organizational psychology and management (Lee and Edmondson, 2017; Bernstein et al., 2016). The strategies we describe that help Spotify address autonomy at scale are per se not new. Although some strategies, such as embedding, might have not been that well documented and the special focus on the large scale might make the collection of practices less common. In our study, we refer to other more traditional environments (Ericsson) and modern companies such as (Atlassian, Zappos), which also report similar approaches. This means that our findings could be useful in large industrial contexts with a high level of adaptability and fast-changing environment, and even open-source projects, as pointed out by related research (Bernstein et al., 2016).

CRediT authorship contribution statement

Darja Šmite: Conceptualization, Methodology, Investigation, Data curation, Formal analysis, Writing – original draft. **Nils Brede Moe:** Conceptualization, Methodology, Investigation, Writing — review & editing. **Marcin Floryan:** Conceptualization, Resources, Validation, Writing – review & editing. **Javier Gonzalez-Huerta:** Data Curation, Formal analysis, Writing – review & editing. **Michael Dorner:** Data Curation, Formal analysis, Visualization, Writing – review & editing. **Aivars Sablis:** Data curation, Formal analysis, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

We sincerely thank the Spotify squads for volunteering their time to participate in our study. This research is co-funded by the Swedish Knowledge Foundation within the ScaleWise project (KK-Hög grant 2019/0087) and the SHADE project (KK-Hög grant 2017/0176), and by the Research Council of Norway through the 10xTeams project (grant 309344).

References

Bernstein, E., Bunch, J., Canner, N., Lee, M., 2016. Beyond the holacracy hype. Harv. Bus. Rev. 94 (7/8), 38–49.
Bungay, S., 2011. The Art of Action: How Leaders Close the Gaps Between Plans, Actions, and Results. Nicholas Brealey Publishing.
Burns, T., Stalker, G.M., 1961. The Management of Innovation. Tavistock, London.
Cantor, M., MacIsaac, B., Mannan, R., 2016. Steering software development workflow: Lessons from the internet. IEEE Softw. 33 (5), 96–102.
Caroli, P., Caetano, T., 2020. Fun retrospectives: activities and ideas for making agile retrospectives more engaging. Available online: https://www.funretrospectives.com/daki-drop-add-keep-improve/.
Cohen, S.G., Bailey, D.E., 1997. What makes teams work: Group effectiveness research from the shop floor to the executive suite. J. Manage. 23 (3), 239–290.
Dingsøyr, T., Moe, N.B., Fægri, T.E., Seim, E.A., 2018. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empir. Softw. Eng. 23 (1), 490–520.
Drach-Zahavy, A., 2004. Exploring team support: The role of team's design, values, and leader's support. Group Dyn.: Theory Res. Pract. 8 (4), 235–252.
Elbanna, A., 2010. Rethinking IS project boundaries in practice: A multiple-projects perspectve. J. Strateg. Inf. Syst. 19 (1), 39–51.
Fowler, M., 2006. Code ownership. https://martinfowler.com/bliki/CodeOwnership.html.
Grant, A.M., Parker, S.K., 2009. Redesigning work design theories: the rise of relational and proactive perspectives. Acad. Manage. Ann. 3 (1), 317–375.
Guzzo, R.A., Dickson, M.W., 1996. Teams in organizations: Recent research on performance and effectiveness. Annu. Rev. Psychol..
Hackman, J.R., 1986. The psychology of self-management in organizations. In: Pallack, M.S., Perloff, R.O. (Eds.), Psychology and Work: Productivity, Change, and Employment. American Psychological Association, Washington, DC.
Ingvaldsen, J.A., Rolfsen, M., 2012. Autonomous work groups and the challenge of inter-group coordination. Hum. Relat. 65, 861–881, 2012.
Jamshidi, P., Pahl, C., Mendonça, N.C., Lewis, J., Tilkov, S., 2018. Microservices: The journey so far and challenges ahead. IEEE Softw. 35 (3), 24–35.
Janz, B.D., Colquitt, J.A., Noe, R.A., 1997a. Knowledge worker team effectiveness: The role of autonomy, interdependence, team development, and contextual support variables. Pers. Psychol. 50 (4), 877–904.
Janz, B.D., Wetherbe, J.C., Davis, G.B., Noe, R.A., 1997b. Reengineering the systems development process: The link between autonomous teams and business process outcomes. J. Manage. Inf. Syst. 14 (1), 41–68.
Kalliamvakou, E., 2017. Collaboration Via Aligned Autonomy for Commercial Software Teams (Doctoral dissertation).
Langfred, C.W., 2007. The downside of self-management: A longitudinal study of the effects tf conflict on trust, autonomy, and task interdependence in self-managing teams. Acad. Manage. J. 50 (4), 885–900.
Lee, M.Y., Edmondson, A.C., 2017. Self-managing organizations: Exploring the limits of less-hierarchical organizing. Res. Organ. Behav. 37, 35–58.
Mankins, M., Garton, E., 2017. How Spotify Balances Employee Autonomy and Accountability. Harvard Business Review.
Manteli, C., Van Den Hooff, B., Van Vliet, H., 2014. The effect of governance on global software development: an empirical research in transactive memory systms. Inf. Softw. Technol. 56 (10), 1309–1321.
Moe, N.B., Šmite, D., Paasivaara, M., Lassenis, C., 2021. Finding the sweet spot for organizational control and team autonomy in large-scale agile software developent. J. Empir. Softw. Eng. 26 (5), 101. http://dx.doi.org/10.1007/s10664-021-09967-3.
Olsson, H.H., Bosch, J., 2016. No more bosses? In: International Conference on Product-Focused Software Process Improvement. Springer, Cham, pp. 86–101.
Paasivaara, M., Lassenius, C., 2014. Communities of practice in a large distributed agile software development organization - case ericsson. Inf. Softw. Technol. 56, 1556–1577, 2014.
Paasivaara, M., Lassenius, C., Heikkila, V.T., 2012. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE, New York, pp. 235–238, 2012.
Petersen, K., Wohlin, C., 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empir. Softw. Eng. 15, 654–693, 2010.
Rey, C., Pitta, N., Ramonas, D., Sotok, P., 2019. Agile purpose: Overcoming bureaucracy. In: Purpose-Driven Organizations. Palgrave Macmillan, Cham, pp. 75–86.
Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons.
de Sitter, U., Den Hertog, J.F., Dankbaarl, B., 1997. From complex organizations with simple jobs to simple organizations with complex jobs. Hum. Relat. 50 (5), 497–534.
Šmite, D., Moe, N.B., Floryan, M., Levinta, G., 2020. Spotify guilds: When the value increases engagement, engagement increases the vlue. Commun. ACM 63 (3), 56–61.
Smite, D., Moe, N.B., Levinta, G., Floryan, M., 2019. Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizatons. IEEE Softw. 36 (2), 51–57.
Šmite, D., Moe, N.B., Šāblis, A., Wohlin, C., 2017. Software teams and their knowledge networks in large-scale software developent. J. Inf. Softw. Technol. 86, 71–86.
Stray, V., Gundelsby, J.H., Ulfsnes, R., Moe, N.B., 2002. How agile teams make objectives and key results (OKRs) work. In: Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering. pp. 104–109.
Sutherland, J., Schwaber, K., 2013. The scrum guide. In: The Definitive Guide to Scrum: The Rules of the Game. Scrum. Org, p. 268.

Darja Smite is a Professor of Software Engineering at Blekinge Institute of Technology, Sweden and a part-time research scientist at SINTEF in Norway. She leads research efforts on global software development and more recently remote working from home. Her research interests include large-scale agile software development and software process improvement. Šmite received a Ph.D. in computer science from the University of Latvia. She has led a number of nationally funded research projects related to the effects of offshoring and scaling for the Swedish software industry, with partners such as Ericsson, Spotify, Telenor, ABB, DXC, Emerson Process Management, and Boss Media.

Nils Brede Moe is a chief scientist at SINTEF im Norway. He works with software process improvement, intellectual capital, innovation, autonomous teams, agile and global software development, and digital transformation. He has led several nationally funded software engineering research projects covering organizational, sociotechnical, and global/distributed aspects. Moe received a dr.philos. in computer science from the Norwegian University of Science and Technology, and holds an adjunct position at the Blekinge Institute of Technology in Sweden.

Marcin Floryan is the Technology Operations Lead at Spotify. He is a passionate technology leader focused on creating an environment where people can do their best work together. Floryan is interested in complex adaptive systems, agile software development, leadership, and organizational development. Floryan received an M.Sc. in biomedical engineering from the Warsaw University of Technology, Poland.

Javier Gonzalez Huerta is an associate professor in Software Engineering at the Blekinge Institute of Technology, in Sweden, working on making companies more effective when managing their software assets aiming at helping them to avoid asset degradation and Technical Debt. Gonzalez-Huerta received his Ph.D. in Computer Science from the Universitat Politècnica de Valencia (UPV) in 2014, after working in the industry for more than 10 years. His research focuses on Asset Management, Asset Degradation, and Technical Debt. Gonzalez-Huerta has been doing applied research together with industrial partners like Ericsson, Spotify, Fortnox, and Swedbank for the last five years.

Michael Dorner is researcher and doctoral candidate at Blekinge Institute of Technology, Sweden. His primary research is to measure, leverage, and improve the continuous exchange and flow of information within communication networks in software engineering and to understand the communication networks' capabilities to cache, transport, deliver, and diffuse information on time. Before academia, he was data scientist and software engineer at Siemens.

Aivars Sablis is a Project Manager in SAF Tehnika JSC, Latvia and a Ph.D. student in Software Engineering at Blekinge Institute of Technology. His research interests include knowledge management and team coordination in large-scale agile software development. He has participated in nationally funded research projects related to global software development and project management in large-scale distributed software development projects, with partners such as Ericsson, ABB, Telia, and Spotify.