Towards Achieving Organizational Agility in Software Development: Identifying Issues and Organizational Alignment Strategies

Nils Messerschmidt
SnT, University of Luxembourg
nils.messerschmidt@uni.lu

Sebastian Schlauderer
University of Bamberg
sebastian.schlauderer@uni-bamberg.de

Sven Overhage
University of Bamberg
sven.overhage@uni-bamberg.de

Abstract

Despite the popularity of agile software development and the existence of agile frameworks that provide methodological support for conducting development projects, companies still often lack an understanding of the changes necessary to achieve organizational agility. This poses the risk that the benefits of agile software development cannot be fully exploited. Building upon Socio-Technical Systems (STS) Theory as analytical lens and the results of 13 semi-structured expert interviews, we identify 13 organizational issues that occur frequently in agile software development. We assign them to categories considering the involved people, technologies, tasks, and structures as constituents of STS, as well as the relationships between them. The resulting scheme of issues provides numerous avenues for the development of additional organizational support that goes beyond the scope of existing agile frameworks. Contributing to this goal, we derive three practical strategies to improve the alignment between the constituents of development organizations from the interviews.

Keywords: Socio-technical systems, organizational agility, agile software development

1. Introduction

In today's rapidly changing business environments, agile software development is becoming ever more popular. When asked about the changes in IT departments in the next five years, decision makers strongly agreed that agile will dominate over traditional ways of working during a recent study (Roth & Heimann, 2023).

Although several agile methods and frameworks exist that provide guidelines for conducting development projects, however, achieving agility is not without challenges. For instance, clients often tend to mimic traditional waterfall projects, as they commonly require fixed budgets, deliverables, and schedules. Other frequently experienced issues when transitioning to agile software development include product owners who do not understand business capabilities, the idea of "only necessary documentation" being misunderstood as "no documentation at all", or endless iterations that do not lead to tangible results (Aghina et al., 2018).

These observations may be partly attributed to the fact that the concept of agility is still perceived ambiguously in practice. Often, it is understood as a way of doing work by following a specific methodology ("doing agile"). To fully leverage the potential of agility, however, it is necessary for companies to achieve *organizational agility,* that is the ability to adapt, change quickly and renew itself ("being agile"). While methodological support for conducting development projects has been established with agile methods and frameworks, an understanding of the changes necessary to achieve organizational agility is still often lacking.

To gain deeper insights into the issues (and possible solution strategies) that arise when trying to achieve organizational agility, we present the results of a study in which we interviewed 13 experts on the topic. To systematically cover relevant parts of development organizations, we used Socio-Technical Systems (STS) Theory as lens of analysis. It focuses on "people", "structure", "task", "technology", and "external environment" as constituents of organizations, and on the relationships between them (Leavitt, 1965). On this basis, we examine the following research question (RQ1): "*Which issues can prevent the achievement of organizational agility and from which constituents of development organizations do they originate?"* To identify solution strategies, we also investigated the potential that arises from a stronger alignment of the organizational constituents. Insufficient alignment of relevant organizational constituents has been highlighted as a potential obstacle to achieving organizational agility before (Hester, 2012; El-Gayar et al., 2013). Accordingly, we examine the following research question (RQ2): *"How can the constituents of development organizations strategically be aligned to increase organizational agility?"*

Our study reveals 13 frequently arising issues, which originate from all constituent elements of development organizations and can prevent the achievement of organizational agility. We also describe three practical alignment strategies to improve organizational agility. The results of our study contribute to the still nascent knowledge base on how to achieve organizational agility. They particularly indicate that there is a need for additional methodological support that focuses on organizational measures and goes beyond the scope of existing agile methods and frameworks.

We proceed as follows: in section 2, we describe the theoretical background. In section 3, we discuss the qualitative research method of our study. The results of our study are presented in section 4. We discuss the results, the implications for academia and practice, and limitations in section 5. The paper ends with a conclusion and an outlook on future research (section 6).

2. Background and related work

2.1. Agile software development

Agile software development radically changes the way development projects are executed. To support this new way of development, several agile methodologies (e.g., Scrum, Extreme Programming) and frameworks (e.g., Large-Scale Scrum / LeSS, Scaled Agile Framework / SAFe) have been introduced. While they vary considerably with respect to their scope and practices, they all build upon the concept of agility. In essence, agility means to "proactively or reactively embrace change and learn from change while contributing to perceived customer value" (Conboy, 2009, p. 340). To achieve this aim, the agile manifesto formulates 12 principles that serve as guidelines on how to approach software development projects (Beck et al., 2001). While the introduced methodologies and frameworks each rely on their own practices to implement these principles and achieve agility, research has identified four concepts that seem to build a common foundation (Baham & Hirschheim, 2022): incremental design and iterative development, inspect and adapt cycles, working collaboratively, and continuous customer involvement. Together, they form the core of agile methodologies.

Research has found that agile software development can lead to several benefits, among them being increased productivity or better customer requirements meeting as prominent examples (Meckenstock et al., 2022). However, the introduction of agile methodologies can also lead to negative outcomes (Petersen & Wohlin, 2009; Petersen & Wohlin, 2010). Potential downsides include stressed developers (Schlauderer & Overhage, 2013), higher process complexity (Overhage et al., 2011), or insufficient documentation of development work (Behutiye et al., 2022). Often, companies seem to fail to realize the benefits of agile development because they did not achieve a high level of competence in agility (State of Agile, 2018). To leverage the benefits of agile development and avoid potential downsides, companies therefore should go beyond simply adhering to agile methodologies ("doing agile") and strive to achieve organizational agility ("being agile").

2.2. Socio-technical systems theory

To achieve organizational agility, companies must transform their development organizations towards becoming more responsive, adaptable, and resilient in unstable environments. To understand which aspects of development organizations are affected by a transformation to achieve organizational agility, we refer to STS theory. It describes an organization as a combination of a social and a technical part and their *environment*. Each part is split into constituents of their own (see Fig. 1): the social part consists of *people* and (social) *structures*, the technical part comprises *tasks* and *technologies*. The constituents are interrelated and must collaborate to achieve the organization's goals (Appelbaum, 1997).

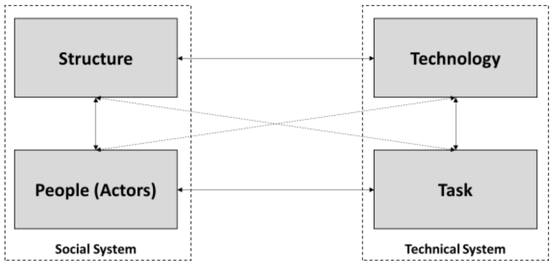

Figure 1. Constituents of socio-technical systems.

A strategic transformation towards achieving organizational agility potentially affects all dimensions of the STS theory. For example, increased stress for developers will impact the people element, while the increased collaboration changes the social structures. Similarly, better meeting of requirements refers to the task of development projects, whereas incremental software deliveries affect the technology category. As a transformation towards achieving organizational agility will hence have socio-technical repercussions, we consider STS theory to be a suitable lens to categorize arising issues and characterize possible solution strategies.

2.3. Related work

With the intensifying discussion on how to achieve organizational agility, research has begun investigating potential issues and organizational challenges. Often, related studies focus on individual constituents of development organizations. With respect to social structures, Weichbrodt et al. (2022) have investigated the role of leadership from a governance perspective. Uwadi et al. (2022) have examined the enabling role of middle management. Kuchel et al. (2023) have focused on the people constituent and identified 15 challenges regarding agile culture. Niva et al. (2023) have focused on an entrepreneurial mindset as enabling factor for successful organizational culture. Regarding the technical constituent, Aldaeej et al. (2023) have assessed a novel approach that addresses the issue of technical debt.

Other studies have emphasized the need for aligning the constituents of development organizations to achieve organizational agility. El-Gayar et al. (2013) have described several issues resulting from insufficient alignment of organizational constituents with illustrative examples. Hester (2012) has shown that system use is impeded when not addressing such imbalances.

So far, however, the knowledge base on how to achieve organizational agility is still nascent and scientific findings are sporadic. Extant research has mainly dealt with issues related to one or a few constituents of development organizations. Approaches to overcome identified issues seem to be even rarer. With our study, we therefore intend to contribute to achieving a more complete picture of the topic.

3. Methodology

To gain deeper insights into the issues and possible solution strategies during a transition to achieve organizational agility, we conducted semi-structured interviews with experts in the field. The experts were gathered from a large German software development company that works for clients across various industries and countries. Each of the interviewees had participated in multiple projects and worked in their respective roles for between fifteen to thirty years. The number of interviewees was determined by the probability of gaining new insights and our goal to sufficiently cover all relevant project roles. Overall, we conducted 13 interviews during August 2023. Roles were chosen according to a typical project setup and included *Business Analysts* (ID1-3), *Delivery Architects* (ID4-5), *Business Architects* (ID6-7), *UX Designers* (ID8-9), *Agile Coaches* (ID10-11), and *Project Managers* (ID12-13). Interviewees were further chosen according to their engagement in relevant communities of practice (e.g., microservices, architecture, global UX design), which have been found to play an essential part for the agile transformation of organizations (Tobisch et al., 2023). The ID is subsequently used for reference purposes in citations.

The interview guideline consisted of two parts. The first part started with questions regarding the experience of the interviewees, such as their background, their education, job experience, and engagement in voluntary communities of practice. We also assessed the interviewees' perception of agility**.** In particular, we asked about the understanding of the interplay of tangible (e.g., practices, methodologies, frameworks) and intangible (e.g., values, principles) agility concepts. Thereafter, we asked the interviewees to describe any issues encountered in typical agile development projects. The second interview part was more of exploratory nature and contained open questions that aimed at discovering beneficial strategies to increase alignment within agile development organizations.

While the first interview part typically was completed in 45 minutes, the duration of the second part varied depending on the availability and knowledge of the interviewee. Accordingly, the total duration varied between 47 to 105 minutes. Each participant was informed that the interview would be recorded, but no details about the content were shared prior to the interview.

After completing the interviews, we transcribed and anonymized the statements. Using STS theory as analytical lens, we then evaluated the statements regarding issues in agile development projects according to Gioia's methodology for inductive research (Gioia et al., 2013). First, we performed open coding using MAXQDA to analyze the statements. We identified 542 in-vivo-codes that point to issues from a practitioner's perspective. In the next step, we performed axial coding. During this step, 405 in-vivo-codes were clustered into 53 higher-level codes referring to issues within the organizational constituents of STS. The remaining 137 in-vivo-codes pointed to diverse issues that affected more than one organizational constituent of STS and could not be further aggregated. Lastly, we performed selective coding. In this step, the 53 higher-level codes were grouped into 13 issue categories and assigned to the organizational constituents of STS they relate to (people, structures, tasks, technology, external environment). The codes referring to cross-cutting issues were assigned to the relationships that connect the involved organizational constituents (task-technology, task-structure, task-people, technology-structure, technology-people, structure-people).

4. Results

4.1. Issues within organizational constituents

Overall, the identified issues could well be mapped to the organizational constituents and their interrelationships as described by STS theory. Next to the intra-organizational constituents, the environment turned out to be an important factor that will therefore be considered specifically in the following. Fig. 2 shows the results of the coding procedure. The highest layer depicts the constituents of STS. The layers below show how 405 of the in-vivo-codes were clustered into issue categories that could be assigned to specific STS constituents. The remaining 137 in-vivo-codes pointed to cross-cutting issues affecting more than one STS constituent. Hence, we refer to them as alignment-related issues. They were assigned to the relationships between STS constituents, which represent potential alignment dimensions and are discussed in section 4.2. Next, we describe the identified issue categories according to the affected STS constituents and provide a subset of representative quotes.

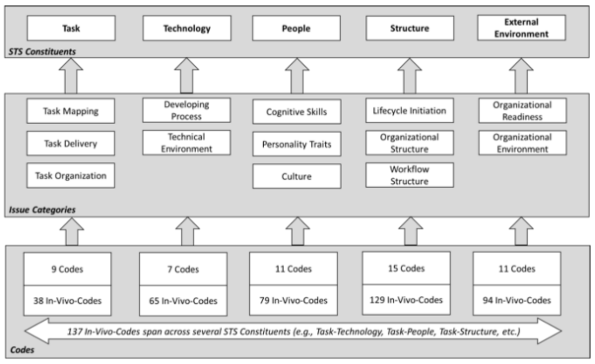

Figure 2. Results of the coding procedure.

Structure-related issues. We found three issue categories related to the structure of organizations: lifecycle initiation, organizational, and agile structure.

A. Lifecycle initiation refers to the early project stages. Before going into development, project teams are assembled with different roles (e.g., architecture specialists, analysts, engagement managers) to set up the foundation of the project and create an appropriate structure. Issues especially include the composition of the set-up team, for instance, a lack of agile experts ("*Scrum masters come into the teams when something is not right […] but the value of the work is not yet seen in the accounts throughout the company.", ID11)*, choosing the wrong team composition, e.g. staffing ("*You must very much look better at how well a character fits into the team. And what does (s)he offer to the team, instead of just thinking, I need another developer, I'll take anyone.", ID8)*, and shoring strategies *("Handling language barrier is a major problem. […] We don't have a translation tool because they're all in the cloud, we have a public sector customer, so we're not allowed to put anything in the cloud, that's a real problem.", ID2).*

B. Organizational structure refers to issues arising from hard factors. They are especially present in larger organizations and would occur in all teams alike. Traditionally, this includes the division of organizational units ("*The gap between IT and business units has become deeper and deeper, so everyone protects him/herself somehow and nobody wants to promise anything.", ID2)*. However, issues can also arise from within the team, through the lack of a unifying agile identity ("*If you act as enterprise architect, then you are in a very overarching position. That doesn't necessarily mean you can't work agile, but you look above the agile projects, you're in a different role.*", ID7). Lacking decision-making competency within teams may further impede agility *("Today it goes through many committees that do not want to give up their competence.", ID6).*

C. Agile structure refers to soft factors that can occur due to employing certain habits when implementing agility. These issues may only affect one or a few teams within a larger project. Common themes are an overly strict adherence to principles *("We were in the daily every day and every 4 weeks we said: now the new sprint will start and then we will be agile. A bit exaggerated, but that's kind of what often happens.", ID1)* and an overinvolvement into agile events ("*In daily stand-up meetings, the whole product team was involved and then at some point, I said that that's no use. Everyone just tells their part, and you really notice how others are already switching off.", ID12).*

People-related issues. We identified three categories: cognitive skills, personality traits, and culture.

A. Cognitive skills are defined as the ability to exert tasks that is gained through education and practice. Especially, agile skills seem to be often lacking *("There are people in teams who say everything used to be better, now I have to deal with a customer myself, I just want to develop.", ID8)*. Some experts also refer to a lack of technical skills ("*It is always like when your 14-year-old son says, let me drive the car, I've been driving with you the whole time, I know how it works. And then you think, no, that's still not a good idea.", ID2).*

B. Personality traits form the counterpart to cognitive skills, as they are not acquirable through education or practice. Mentioned issues especially include a lack of ownership ("*You ask something, there are only four people sitting there […] did you get it? No answer. Do you still have any questions? No answer. So, do you know exactly how to do it? No answer.", ID2)* and a lack of courage *("We are lacking the courage to sometimes make a mistake and knowing that it is not bad. Just to try to implement it.", ID8).*

C. Culture refers to values that are supported by the larger organization, and thus institutionalized. Culture-related issues include the lack of entrepreneurial thinking ("*A startup works in a common vision. […] people who don't have anything to do right now, they just look for another area or another company.", ID8)* and the lack of a supporting system ("*Then there is this blame game […] when somehow the button should be red and the background is also red, that's not my fault. You weren't paying attention.", ID8).*

Technology-related issues. We found two categories: the development process, and the environment.

A. The development process forms the core of the project lifecycle. Several issues can arise when working software has to be implemented in longer lasting and/or complex projects. This includes the code quality ("*The proportion of the work in the sprint that is spent on new functionality will become smaller and smaller and the proportion of maintaining the quality of the existing functionality will become larger and larger.", ID10)* and standardization *("Imagine a huge project, with many different teams, if each one does it a little differently, then you don't get maintainable software.", ID7).*

B. The technical environment into which the software must be embedded is equally important for development success. Concerns particularly relate to external dependencies ("*Mostly with the larger IT systems at the customer, you just have a service, and it has been there for 20 years and yes, you should replace it.", ID8)* and the fit from a strategic perspective *("We build up the entire team with developers, you are starting to build the architecture, the technology must be in place, but you don't even know what actually needs to be done.", ID9).*

Task-related issues. We identified three categories: mapping, delivery, and organization of tasks.

A. Task mapping involves the mapping of business requirements onto tasks the agile teams need to deliver. Issues particularly arise with respect to the granularity of tasks ("*Anticipate what is coming, but at the same time no longer writing everything down in detail, that is still very, very difficult for us.", ID10)* and the freedom of how a task needs to be delivered ("*I know that it would be good to just writing down requirements and leaving the solution path completely open to the developers. My experience is that it just doesn't work.", ID2).*

B. Task delivery can be affected by issues related to the execution of the defined tasks. This includes the synchronization within the team ("*That's always an organizational challenge in the team, keeping an eye on synchronizing the tasks and making it clear who is doing what and with whom.", ID1)* and the depth of delivery *("For instance, when you build a dialog, it is already completely built through, instead of saying I maybe only need a table with 3 columns to do my process.", ID4).*

C. Task organization is affected by the focus on features ("*You have a much stronger feature pressure […] and you get into a really bad vicious circle that's different and worse than in waterfall.", ID4)* and maintenance ("*At some point, we had replanned it based on how it was delivered in the past […] and then we realized that we would never, ever stay within the schedule.", ID8).*

Environment-related issues. Often disregarded in theory, we found the external environment to have a major impact on the success of agile software development projects. We identified two issue categories: organizational readiness, and organizational environment.

A. Organizational readiness includes the factors that influence the success of agile projects from the internal perspective. Despite best agile efforts of some, the organization is often found to remain deeply hierarchical, leading to a major internal disconnect ("*The company decided to only make agile contracts, but the people who sit on the customer side often do not have this mindset yet and are trapped in a very hierarchical organization where steering committees have certain expectations and milestones are set in a 6-month cycle.", ID10).* Organizations often lack the necessary mindset shift to conduct agile projects successfully. Instead, the traditional logic of project success is transferred directly to agile projects ("*What would help if you didn't look at the pure numbers, how many story points they have somehow managed to achieve, but rather the added value that happened. […] Instead of saying, okay, the team manages to deliver 37 story points*.", *ID8*).

B. Organizational environment covers the factors that influence the success of agile projects from the external perspective. Market pressure is found to be one of the key issues, with strict implications on timelines and budget. This clashes with the non-deterministic nature of agile projects ("*If I say I'll do an architecture project […] and after it is finished, you still have nothing countable in the sense of less costs, more sales, because what counts in the end is always the money, that's difficult to justify.", ID6).* This also influences the degree of risk acceptance, leading clients to lack foresighted thinking ("*The customer wants to know when you will be done, and you must be able to make a statement about it without knowing everything. So, the customer must be able to accept this uncertainty and live with it.", ID2*).

4.2. Cross-cutting issues & alignment strategies

During the interviews, we often found concerns about the understanding of agility on various organizational levels: "*I see a lot of poor agility here in our commitments, and also a very poor agile understanding among organizations and also among ourselves*" (ID11). Oftentimes, software development and agile ways of working are also viewed separately: "*Technological problems and problems caused by agility are still strictly separated*" (ID7). However, to become agile on an organizational level, their interdependencies need to be accounted for simultaneously. We found the following pain points that we describe along the six alignment dimensions of STS.

Technology and Structure. Frictions regarding this dimension were frequently recognized in the study. While issues typically manifest in the form of technical errors or failures, the underlying reason often is the organizational structure. According to Conway's Law, the IT is dependent on the organizational structure: "*Conway's Law can be an anti-pattern if, of course, when the organization is bad and you build the IT according to the organization, then you have built bad software*" (ID5). On the other hand, technology can also impede functioning structures, for instance, due to large monoliths that small teams cannot handle efficiently: "*Of course, you don't have to generally smash every monolith. [...] but I haven't seen any large monolith yet that doesn't block agility, simply because there are 10 teams working on such a large piece of software, they break things so often."* (ID7). A lack of consciously shaping communication structures was apparent in the projects: "*The flow of communication should be considered. So, the question is not about the cut of the user stories, because within a team you talk, but rather the cut of the teams. You can do a lot wrong."* (ID5).

Technology and Task. The need to align task and technology has long been recognized in practice and is primarily concerned with the question of which software architecture to employ for a given context. In agile development, it seems to be a challenge how to divide technological issues (e.g., when having a monolith) into manageable and executable tasks, particularly in rapidly changing environments. This may also require taking unpleasant decisions that requires additional work upfront: "*You know that testing is complicated, you don't like technical changes, you don't like replacing frameworks because it's always painful to do*." (ID4). On the other hand, there is a risk of becoming too fine-granular that causes the teams to lose its orientation on value: *"Often, the architects tend to cut it even smaller and if they do that, then a microservice can no longer be assigned to a value and then it becomes difficult."* (ID10).

Technology and People. A major issue within the people-technology dimension is the lack of involvement or even the detachment of various roles from the team, for instance, with respect to the product owners *("There are management issues, if I want a team to organize itself. The managers must lead differently.", ID10)* or enterprise architects ("*What architecture do I need that is able to adapt to the changes, and I think not enough architects are asking themselves this question, but rather they are looking for the optimal architecture for the moment and then move on to the next project. [...] But have no idea whether the stuff they've done works or not, if in doubt, they don't get any feedback.", ID11).*

Task and People. A dimension that is rarely been mentioned with a need to be aligned by the interviewed experts is that of task and people. Nevertheless, it is important to consider, as agile micro-behaviors often seem to be missing: "*As soon as stressful situations come, people have their trained reflexes as they simply learned how to deal with problems. And these are often not agile behaviors, but rather micromanagement, and you must manage that."* (ID2). Limiting people as to how they approach the tasks might also negatively affect the outcome: "*If the developers are not allowed to make suggestions or something similar, because you write a specification more or less with every story, then I always think that is something where you simply lose the commitment of the development team."* (ID5).

Structure and People. Naturally, the organizational structure and people working under it are strongly interrelated: "*If you ask the user, how do you want to work? When he works in Excel today, he will also have an Excel table that explains how he wants it*." (ID9). This calls for agile structures to be explicitly shaped in alignment with the people: "*It doesn't help to work against the people, so it depends very much on the form of organization. What kind of people are there, what are their skills, I don't think there is a panacea*." (ID4).

Structure and Task. Regarding the alignment of structures and tasks, it is being recognized that there is no "one-size-fits-all" agile way of working. Implementing inflexible frameworks commonly leads to 'agile waterfalls' *("Quite clearly, such an architectural decision is important if I want to build something. When it simply cannot be built with 10 people, i.e., in the classic Scrum team, of course that gets in the way of agility.", D7),* or leads to overhead and technical debt that becomes increasingly difficult to resolve ("*What we want to do hasn't really been defined yet. But we've already started and that's also a problem where we as UX designers are always behind, namely that we've already made so many decisions and we must correct them first.", ID9).*

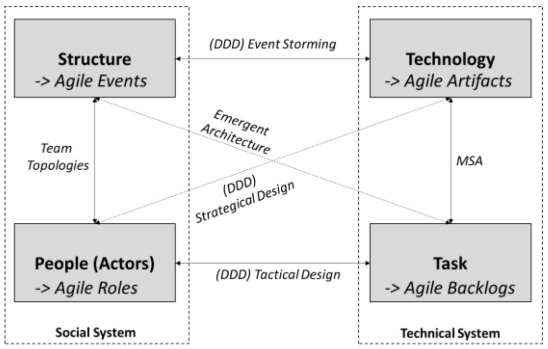

Figure 3. Proposals to improve alignment.

From the interview data, we identified three strategic solutions to improve the alignment in agile development organizations (see Fig. 3).

Microservice Architectures (MSAs) are proposed as alignment method for the technology-task dimension. MSAs are a style of software architecture that emerged in the 2010s in contrast to monolithic architectures. The primary design principles of microservices include loosely coupled and highly cohesive service boundaries that are aligned with business capabilities. Such boundaries enable each service to function on its own, leading to the ability to independently deploy microservices without the need to account for interdependencies to other services (Newman, 2021). The characteristics of MSAs have implications on how technical work is divided and are thus deemed appropriate to improve alignment: "*That's the idea of microservice architectures: If there are new requirements, it is relatively easy to say that something needs to be changed in the software, so ideally everything else can stay as it is.*" (ID1).

Domain-Driven Design (DDD) can be seen as a philosophy that encompasses several software design patterns closely aligned with business capabilities. It was first proposed by Evans (2004). Originally not meant for any software architecture in particular, the potential of DDD was emphasized by the interviewed experts as its patterns can facilitate the process of building microservices. Generally, three types of patterns can be applied to the alignment context with varying scope. These are (from highest scope to lowest):

Event Storming. To address the structure-technology gap, it is proposed to apply the event storming pattern as specified by Brandolini (2013). This is a powerful tool addressing the entire domain of the organization, i.e., what the organization does and the business environment it operates in (e.g., banking, logistics, health, insurance). At this stage, non-technical and technical personnel (e.g., domain experts, product owners, technical developers) come together to conduct a workshop to explore and refine the business domain: "*We introduced event storming to make the microservice cut into several pilot projects, that works very well.*" (ID7).

Strategical Design. The strategical design, situated at the sub-domain level, addresses the people-technology-dimension. One overarching domain can be distinguished into different sub-domains (e.g., invoicing, loyalty programs). At this stage, the patterns are applied to define bounded contexts with ubiquitous languages that are to be used within the same teams: "*The advantage is that the business is involved, that you really have to agree on one language as a team, including business and IT, that's definitely an advantage*." (ID13).

Tactical Design. The alignment of people and tasks can be improved through the tactical patterns, which are situated at the lowest and most granular level of DDD. For each sub-domain, a specific approach for implementation needs to be chosen. The tactical patterns are set to increase maintainability and adaptability to changing requirements and can be employed off-the-shelf: *"If I know that we have shared common, consistent rules that should be adhered to, and if this is in agreement with the department, it's a great idea. Especially when introducing new rules that I can think of later, this will make the change more maintainable locally.*" (ID5).

Team Topologies (TT) was mentioned as a third approach. It is a best practice approach developed by Skelton and Pais (2019) to design software development teams, aiming to address the disconnect along the structure-people dimension. By adhering to the team distinctions, the "inverse Conway maneuver" (Fowler, 2022) is performed, referring to the structuring of organizations according to their intended architectural design.

Two basic patterns are developed to optimize the development flow: The way of working is distinguished along four types (e.g., stream-aligned, enabling, complicated subsystem, platform), as well as the interaction between teams along three variants (e.g., facilitating, X-as-a-Service, direct). According to the experts, this seems to be a promising approach to shaping structures according to agile people's needs: "*Teams interact vertically and horizontally. Up to what size is such a team feasible? We must think about it again and again. [...] And it doesn't stop there, new people come into the project, and then you notice, we don't have this structure anymore because the people who came in just work differently. This is a theme that is always present, and that's why I find the TT approach exciting*." (ID9). In initial trials, the TT approach has been perceived as successful: "*I also talked about this human overload, or the overload of teams, and team topologies can help. Because not every team can do everything, we have a team which is concerned with platform issues, which supports other teams and takes on cross-sectional tasks.*" (ID4).

Recognizing the value of an emergent architecture is found to avoid framework lock-in effects with organizational overhead. Therefore, a semi-agile approach, by adhering to the principle of shaping the architecture upfront as much as needed but allowing for other elements to emerge naturally over the project lifecycle, is found to retain a higher flexibility, allowing to revisit technical prerequisites based on changing business requirements: *"You must look at it per account or per organizational context and then derive activities from there. What we need is a sort of construction kit and then I must decide for each context at each point in time: What makes sense? Which method work best?"* (ID3). Vice versa, by shaping structures only as much as needed, this allows agile events to be shaped and continuously evaluated to the needs of the product teams and, thus, limits the structural overhead often observed in scaled frameworks.

5. Discussion

5.1. Key findings

The results of our study show that several organizational issues can make the road to achieving organizational agility difficult. This shows that issues can originate from all constituents of organizations, that is from the involved people, tasks, structures, technologies, and the external environment. In addition, issues arise if the constituents of development organizations are not well aligned to achieve organizational agility. By identifying organizational issues and solution strategies across all constituents of development organizations, our study contributes to the still nascent knowledge base on how to achieve organizational agility in several ways.

Firstly, despite growing scholarly interest in scaled agile frameworks (e.g., Theobald et al., 2019; Edison et al., 2021) our assessment suggests that current scaled agile solutions are not capable of fully satisfying the needs of agile organizations. In particular, no scaled agile framework explicitly considers the internal and external organizational environments. Generally, two groups of scaled agile frameworks can be distinguished: Scrum-related frameworks (e.g., Scrum of Scrums, Nexus, LeSS) represent the social system, especially the structural category, more strongly. Other approaches such as basic Kanban and the more abstract SAFe offer stronger guidance with respect to the technical subsystem, particularly regarding the task category. LeSS and SAFe are capable to address many of the issues identified. While LeSS is strongest in the people category, SAFe performs better in the structural category. However, both scaled agile frameworks cannot completely satisfy organizational needs as neither addresses the external environment. Moreover, the alignment dimensions discussed by the interviewed experts seem to be completely disregarded by current frameworks. Going beyond the scope of these frameworks, the results of our study provide guidance on how the alignment of organizational constituents can be improved.

While traditional client-server-architectures possess advantages in terms of development, testing and deployment costs, the components are deeply interrelated and costly to break apart. Updating the functionality means redeploying a new version of the entire application. Similarly, monoliths enable only horizontal scaling, thus replicating the entire monolith on a new server. This is a hazard especially for cloud migration purposes (Lewis & Fowler, 2014), which remains a growing topic to many companies. Thus, implementing modular, component-based architectures possess major advantages that can mitigate some of these issues. Other possible approaches within that space are service-oriented architectures, which are found to better fit complex infrastructural issues that involve the entire enterprise, or self-contained systems, that are found easier to manage from a structural point of view (Cerny et al., 2017).

Closely linked to the surge of interest in microservice architectures is the recognition of DDD. While research has long neglected DDD, more recently, several publications have addressed the potential to link DDD concepts with modern software architectures, thus supporting their feasibility to improve the alignment in development organizations. Such links include designing microservices (Hippchen et al., 2017) and identifying microservice boundaries (Vural & Koyuncu, 2021).

The team types and interactions within the team topologies (TT) approach are well-known in industry. Leite et al. (2021) identified four organizational structures and found they are strongly related to the TT patterns. Despite being widely recognized in industry, scientific references remain sparse, apart from a few recent contributions. Ahmed and Colomo-Palacios (2021) confirmed the basic patterns of the TT approach based on a systematic literature review. The TT approach, along with multiple other organizational taxonomies developed by an active community of software engineering researchers (Tamburri et al., 2022), will help practitioners move towards consciously shaping team structures.

In this study, a novel approach was presented to align organizational dimensions. The interviews have led to the proposition of several alignment strategies that enable organizations to more fully leverage the benefits of agile development. Nevertheless, it is not enough to merely adhere to agile practices and frameworks to become agile. The notion of "being agile" versus "doing agile" is, as seen in the qualitative interviews, a relevant distinction to make. Past research on agility has mainly focused on practices and frameworks. There is a need to more strongly examine how the values and principles of agility can be spread across large organizations to avoid a state in which the different parts of the organization become disconnected by different speeds.

5.2. Implications

The results of our study have implications for academia and practice. As regards academia, they signal the need for further research as several issues remain when transitioning to organizational agility. We could show that the STS theory (Leavitt, 1965), while being developed long before agility, continues to capture relevant issues today. Compared to previous research linking the STS theory to agility (e.g., Stanley & Aggarwal, 2023; Kolukuluri & Singh, 2023), we present a holistic and extensive coding scheme of issues identified in agile ways of working. Of the over 500 in-vivo codes identified, more than 400 can be directly mapped against one of the five constituents relevant to this study.

It was further found that issues in software development can be accounted for through the STS lens, particularly cognitive load (e.g., Skelton & Pais, 2019), team boundaries or responsibilities (e.g., Espinosa et al., 2003), and establishing informal and formal communication flows (e.g., Stapel & Schneider, 2014). Research is encouraged to further explore these concepts and their alignment in conjunction with the STS theory.

As discussed above, current scaled agile frameworks fall short of solving all the issues identified in this study. Applying scaled agile frameworks was commonly found to leave organizations with structural overhead, whereas the flexibility and value-driven purpose of lean agility got lost to some extent. To date, no unifying holistic agile framework that captures the identified issues in its entirety exists, and the alignment proposals discussed here are merely limited to best practice usages. This calls for the creation of new agile frameworks, or the extension of established ones, to support the transitioning towards organizational agility.

For practice, façade agility remains a major issue. Practice is therefore encouraged to target agility, and particularly the agile mindset of all employees in the organization. Often, agile practices are either not understood correctly, or do not fit the underlying organizational structure (e.g., due to having subtle hierarchical structures that inhibit the exertion of agility).

This research further provides an extensive scheme with potential obstacles that ought to be observed when transitioning to achieve organizational agility. While any organizational strategy will need to consider the context, size and scope of the agile software development project, the presented alignment strategies can serve as (a part of) a toolbox of recommendations while transitioning towards organizational agility.

5.3. Limitations

We have taken several precautions to ensure the validity of our results. In addition to carefully selecting experts with extensive experience, we decided to conduct semi-structured interviews as these lead to more comparable results. Nevertheless, there are several limitations in the light of which our results should be interpreted. Although we followed an interview guideline, the external validity of our results could be affected by interviewer bias and the rather small sample size. Our findings would therefore benefit from validation in future studies. Moreover, we have so far only investigated solution strategies that aim at a better alignment of organizational constituents. Since there are likely other strategies, our results are by no means exhaustive but rather contribute to a growing knowledge base. They should hence be complemented by additional studies.

6. Conclusion

Achieving organizational agility is important to fully leverage the potential of agile software development. Using STS theory as lens of analysis, we identified 13 categories of issues that can arise in agile software development and need to be addressed to achieve organizational agility (RQ1). Our findings indicate that it is not sufficient to merely follow agile procedures to mitigate these issues. Rather, it seems necessary to induce more profound organizational changes. This endeavor can be facilitated by providing additional methodical support that goes beyond the scope of current agile frameworks. Therefore, we derived three practical strategies to improve alignment between the different constituents of development organizations (RQ2).

The results of our study provide avenues for future research. Future research could, for instance, verify the identified issues in quantitative studies and/or develop additional methodical support to solve them. For practice, our results provide a scheme with potential obstacles that should be monitored when transitioning to organizational agility. Depending on the situation, the presented alignment strategies might provide a means to make agile development organizations more effective. Future research should complement them to provide a more holistic methodological support. We hope that our results can provide a starting point for such endeavors.

Acknowledgements This research was funded in part by the Luxembourg National Research Fund (FNR) and PayPal, PEARL grant reference 13342933/Gilbert Fridgen. For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

Aghina W, Handscomb C, Krishnakanthan K, Thaker S (2018) 10 common myths about enterprise agility. McKinsey & Company. http://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/the-organization-blog/10-common-myths-about-enterprise-agility. Accessed 2023-03-23.

Ahmed W, Colomo-Palacios R (2021) Team Topologies in Software Systems: A Multivocal Literature Review. In: International Conference on Computational Science and Its Applications, pp 272-282.

Aldaeej A, Nguyen-Duc A, Gupta V (2023) A Lean Approach of Managing Technical Debt in Agile Software Projects– A Proposal and Empirical Evaluation. In: International Conference on Agile Software Development, pp. 67-76.

Appelbaum SH (1997) Socio-technical systems theory: an intervention strategy for organizational development. Management decision 35(6):452-463.

Baham C, Hirschheim R (2022) Issues, challenges, and a proposed theoretical core of agile software development research. Information Systems Journal, 32(1):103-129.

Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Mariack B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) The agile manifesto. https://agilemanifesto.org/. Accessed 2024-09-04.

Behutiye W, Rodríguez P, Oivo M, Aaramaa S, Partanen J, Abhervé A (2022) Towards optimal quality requirement documentation in agile software development: A multiple case study. Journal of Systems and Software, 183(1):1-17.

Brandolini A (2013) Introducing event storming. https://ziobrando.blogspot.com/2013/11/introducing-event-storming.html. Accessed 2024-09-04.

Cerny T, Donahoo MJ, Pechanec J (2017) Disambiguation and Comparison of SOA, Microservices and Self-Contained Systems. In: Proc International Conference on research in adaptive and convergent systems, pp 228-235.

Conboy K (2009) Agility from first principles: Reconstructing the concept of agility in information systems development. Information systems research, 20(3):329-354.

Edison H, Wang X, Conboy K (2021) Comparing methods for large-scale agile software development: A systematic literature review. IEEE Transactions on Software Engineering, 48(8):2709-2731.

El-Gayar O, Sarnikar S, Wahbeh A (2013) On the design of IT-enabled self-care systems: A socio-technical perspective. In: Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS), pp 2484-2493.

Espinosa JA, Cummings JN, Wilson JM, Pearce BM (2003) Team boundary issues across multiple global firms. Journal of Management Information Systems, 19(4):157-190.

Evans E (2004) Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional.

Fowler M (2022) Conway’s Law. https://martinfowler.com/bliki/ConwaysLaw.html. Accessed 2023-10-21.

Gioia DA, Corley KG, Hamilton AL (2013) Seeking qualitative rigor in inductive research: Notes on the Gioia methodology. Organizational Research Methods, 16(1):15-31.

Hester AJ (2012) Measuring alignment within relationships among socio-technical system components: A study of wiki technology use. In: Proceedings of the 50th annual conference on Computers and People Research, pp 147-154.

Hippchen B, Giessler P, Steinegger R, Schneider M, Abeck S (2017) Designing microservice-based applications by using a domain-driven design approach. International Journal on Advances in Software, 10(3):432-445.

Kolukuluri M, Singh JB (2023) A qualitative study on project failure in agile teams using socio-technical systems theory. In: Proceedings of the 56th Hawaii International Conference on System Sciences (HICSS), pp 6528-6537.

Kuchel T, Neumann M, Diebold P, Schön EM (2023) Which challenges do exist with agile culture in practice? In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. pp. 1018-1025.

Leavitt HJ (1965) Applied organisational change in industry: Structural, technological and humanistic approaches. In: March JG (Ed.), Handbook of organisation. Rand McNally, Chicago, Illinois, pp 1144-1170.

Leite L, Pinto G, Kon F, Meirelles P (2021) The Organization of Software Teams in the Quest for Continuous Delivery: A Grounded Theory Approach. Information and Software Technology 139(1):1-19.

Lewis J, Fowler M (2014) Microservices – a definition of this new architectural term. http://martinfowler.com/articles/microservices.html. Accessed on 2023-05-07.

Meckenstock JN, Hirschlein N, Schlauderer S, Overhage S (2022) The business value of agile software development: Results from a systematic literature review. In: Proceedings of the 30th European Conference on Information Systems (ECIS), pp 1-17.

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

Niva P, Paasivaara M, Hyrynsalmi S (2023) Striving for Freedom in a Large-Scale Agile Environment with an Entrepreneurial Mindset of a Product Owner. In: International Conference on Agile Software Development, pp 97-114.

Overhage S, Schlauderer S, Birkmeier D, Miller J (2011) What makes IT personnel adopt scrum? A framework of drivers and inhibitors to developer acceptance. In Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), pp 1-10.

Petersen K, Wohlin C (2009) A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case, Journal of Systems and Software 82(9):1479-1490.

Petersen K, Wohlin C (2010) The Effect of Moving from a Plan-Driven to an Incremental Software Development Approach with Agile Practices, Empirical Software Engineering 15(6):654-693.

Roth SL, Heimann T (2023) IT-Trends 2023 Study. Capgemini, Berlin.

Schlauderer S, Overhage S (2013) Exploring the Customer Perspective of Agile Development: Acceptance Factors and On-Site Customer Perceptions in Scrum Projects. In: Proceedings of the International Conference on Information Systems (ICIS), pp 1-20.

Skelton M, Pais M (2019) Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution.

Stanley DS, Aggarwal V (2023) Framework for enablers and outcomes of workforce agility: socio technical systems perspective. International Journal of Business Excellence 30(4):490-506.

Stapel K, Schneider K (2014) Managing knowledge on communication and information flow in global software projects. Expert Systems, 31(3):234-252.

State of Agile (2018) 13th annual state of agile report. http://stateofagile.com. Accessed on 2023-05-26.

Tamburri DA, Kazman R, Fahimi H (2022) On the Relationship Between Organizational Structure Patterns and Architecture in Agile Teams. IEEE Transactions on Software Engineering, 49(1):325-347.

Theobald S, Schmitt A, Diebold P (2019) Comparing scaling agile frameworks based on underlying practices. In: International Conference on Agile Software Development, pp 88-96. Springer, Cham.

Tobisch F, Schmidt J, Matthes F (2024) Investigating Communities of Practice in Large-Scale Agile Software Development: An Interview Study. In: International Conference on Agile Software Development, pp 3-19.

Uwadi M, Gregory P, Allison I, Sharp H (2022) Roles of middle managers in agile project governance. In: International Conference on Agile Software Development, pp. 65-81.

Vural H, Koyuncu M (2021) Does domain-driven design lead to finding the optimal modularity of a microservice? IEEE Access 9(1):32721-32733.

Weichbrodt J, Kropp M, Biddle R, Gregory P, Anslow C, Bühler, UM, Mateescu M, Meier A (2022). Understanding Leadership in Agile Software Development Teams: Who and How? In: International Conference on Agile Software Development, pp 99-113.