Organizational Conflicts in the Adoption of Continuous Software Engineering

Eriks Klotins[1(✉)] and Elliot Talbert-Goldstein[2,3]

[1] Software Engineering Research Lab (SERL), Blekinge Institute of Technology, Karlskrona, Sweden
eriks.klotins@bth.se

[2] Empirical and Applied Software Engineering Lab (EASEL), University of Maryland, Baltimore County, USA
el3@umbc.edu

[3] Bosserman Center for Conflict Resolution, Salisbury, USA

Abstract. Software is a critical component of nearly every product or service. Improvements in software can lead to substantial competitive advantages. At the same time, software and surrounding engineering teams have become increasingly complex.

The adoption of continuous integration and delivery is a recent trend to radically improve software release speed. However, its adoption is far from straightforward. Specifically, rethinking processes, organizational culture, ways of working, and business models require buy-in from diverse stakeholders that may have conflicting objectives. Such situations are explored by organizational conflict research.

This paper reports on early lessons from an ongoing research project in continuous software engineering, specifically investigating adoption challenges from an organizational conflict perspective. We identify catalysts, symptoms, and outcomes of organizational conflicts hindering the adoption process.

We conclude that predictable conflicts emerge when adopting continuous engineering. Engineers, managers, and other teams can proactively prepare for and allocate resources to resolve them. Proper analysis and management can help avoid wasted time, impeding processes, and frustration.

Keywords: Continuous Software Engineering · Organizational conflicts · Change management

1 Introduction

In this paper, we report on an ongoing research project into the adoption of continuous software engineering (CSE). Our focus is continuous integration and delivery, however we also consider broader organizational implications, such as planning and requirements on collaboration [6].

Continuous software engineering is a set of principles promoting rapid and frequent delivery of incremental software updates and extensive use of feedback to steer development [6]. Observing changes in product usage patterns due to

C. J. Stettina et al. (Eds.): XP 2023, LNBIP 475, pp. 149–164, 2023.
https://doi.org/10.1007/978-3-031-33976-9_10

software changes informs engineers about the success of changes and support further product decisions [7,11]. However, the key differentiating characteristic of CSE is the immediate delivery of any changes by developers to the end-users. This speed is achieved by extensive automation of build, test, integration, and delivery steps and by removing organizational bottlenecks.

Companies aiming to improve their software engineering processes by adopting continuous engineering principles face major challenges. Partly, the challenges are associated with adopting new tools and technologies. However, earlier studies suggest that the required organizational changes are the most significant challenge by far [2,10,11].

Deeper organizational inefficiencies are unlikely to be solved with new technologies and automation tools. Specifically, misunderstandings, misalignment, and sub-optimal organizational structures will limit the benefits of adopting process automation [6,10].

Misalignment within and across organizations is addressed by organizational conflict research. Foundational research on organizational conflict and software engineering includes structural and interpersonal issues within and between organizations [1,8,12,15,21].

Studies of continuous integration and delivery primarily address social and technological obstacles. Such as challenges associated with team members changing roles and learning new skills and tools. However, only some, if any, focus on challenges establishing an efficient cross-organizational coordination and collaboration [3,5,13,14]

We aim to understand to what extent underlying organizational conflicts hinder the adoption of CSE. We use three industrial cases to explore the adoption challenges and symptoms pointing toward deeper organizational inefficiencies.

Our results suggest that common adoption challenges, such as stakeholder resistance, restrictive processes, and organizational silos, can be explained by underlying organizational conflict. We conclude that organizations need to improve their conflict resolution approaches and advocate for continuous conflict management as part of the adoption strategy.

2 Background and Related Work

2.1 Organizational Conflicts in Software Engineering

In its most basic form, an organizational conflict is a misunderstanding or disagreement, real or perceived, around the needs, interests, and values of people working together [9].

CSE is a cross-cutting phenomenon requiring cooperation between all parts of an organization. The multitude of stakeholders and potential differences among them create particular causes of conflicts that need to be explored in the context of CSE [11,16].

The causes of organizational conflict can be broken out into six broad categories: task interdependence, goal incompatibility, ambiguous rules, differences in values and beliefs, resource scarcity, and ineffective communication [15]. These causes manifest differently in different scenarios, and specific components become more important for different types of conflicts.

Jehn [8] introduces an empirical typology of organizational conflicts. The topology covers the negative and positive impacts of conflict and incorporates both “management teams” and “production groups.” The three key types of organizational conflict are task, relationship, and process. Task conflict occurs when team members may disagree on how to perform work. The research indicates moderate task conflict can be constructive and stimulate discussion of ideas that help groups perform better and group productivity. However, relationship conflict, where teams have chronic issues that lead members to be “negative, irritable, suspicious, and resentful,” have significant adverse effects on productivity. Process conflict connects the two other types. Conflicts over *process* include how tasks should be accomplished, who is responsible, and how to delegate responsibility. Lower levels of process conflict were shown to limit the negative impacts on a team’s performance. This tripartite typology has become a foundation for analyzing team conflict, especially in software engineering.

Further studies have extended Jehn’s [8] work to include intergroup conflict [12,21]. In software engineering specifically, this typology has been applied and empirically corroborated [9,21,22]. Enterprise software adoption research includes social network analysis to evaluate the impact of the propagation of intergroup conflict including issues with bureaucracy, interpersonal problems, and conflicting corporate values [21].

Siddique et al. [20] investigates the causes, consequences, and mitigation strategies of conflicts from the perspective of project managers. The causes include concerns like the role and knowledge of the product owner or customer, organizational hierarchy, bureaucracy, contracting and finances, and personal egos. Consequences included problems with productivity, wasted time and distractions, poor decisions, and lack of communication. The research covered possible strategies for addressing conflict, focusing on root-cause analysis as the primary means to resolve conflicts.

Power [17] introduces tools to study and remove impediments in Agile projects using impediment impact diagrams. Impediments included categories that comprised technical issues as well as interpersonal issues. The diagrams allow team members to compare the benefits of implementing a specific effort to remove an impact and the internal or external resources needed to do it.

Studying the types of conflicts that may occur during the adoption of CSE is not a direct route. It is essential to look at conflicts in teams and organizations generally and during organizational change. Additionally, it is helpful to narrow the scope of possible conflicts to software engineering and Agile contexts, which provide the foundations for CSE.

2.2 Continuous Software Engineering

Continuous software engineering originates from lean and agile principles. The overarching goal of CSE is to ensure a continuous flow of software from its inception to creating value for the user [6].

The continuous flow of software is achieved by delivering software in small increments and automating integration, verification and delivery steps. Frequent and small software deliveries creates an opportunity to gauge the success of the update by collecting focused telemetry and feedback [7,11].

The benefits of CSE include the flexibility to react rapidly to new market opportunities and leverage data-driven decision-making. An automated pipeline allows engineers to focus on value-adding tasks and offload menial tasks to automation [6,7]. The adoption process requires the organization to rethink its ways of working, set specific goals, and promote continuous improvements throughout the organization and its heterogeneous teams [10].

Implementing a CSE pipeline in an existing organization is a substantial undertaking. Adopting new tools, automation, test data preparation, and pipeline maintenance requires substantial investments. Furthermore, successful adoption of CSE requires cross-cutting changes in the organization [2,11,16]. For instance, to support the rapid delivery of software updates, the decisions of what to deliver need to be made rapidly. Fast planning has an upstream dependency on flexible resource planning and organizational support. Delivery of frequent updates requires buy-in from downstream stakeholders and customers. Such changes may require renegotiating customer contracts and tailoring business models [7,11].

Our work with several industrial partners highlights that aligning the goals of different parts of an organization remains one of the key challenges in adopting continuous engineering. Misalignment often stems from different interpretations of the same goal. For example, improving efficiency may be interpreted by an R&D department as delivering more new features. For operations, efficiency may mean spending fewer resources on providing stable services. More frequent software updates increase the risk of disrupting smooth operations and are at odds with stability. In such situations, organizational conflicts may emerge and hinder attempts to improve organizational performance [10].

3 Research Methodology

The aim of our research is to explore how organizational conflict research can support organizations and practitioners in adopting CSE. To guide our study we define the following research questions:

RQ1: What are symptoms of organizational conflict in the context of adopting CSE?

Motivation: With this research question we want to explore to what extent difficulties in adopting CSE can be attributed to hidden and/or unaddressed organizational conflicts.

RQ2: What are the root causes of identified symptoms from an organizational conflict perspective?

Motivation: With this research question we want to explore the underlying causes for organizational conflict and map them to conflict resolution strategies.

RQ3: What are the advantages of studying conflicts in CSE empirically?

Motivation: Most of research focuses on tooling and technical aspects of CI/CD. However, a degree of organizational streamlining are required to enable end-to-end automation. With this research question we aim to clarify how studying organizational conflict can support adoption of CSE.

3.1 Research Approach

We conduct this study as part of an ongoing industry-academia research project into adopting continuous software engineering in the industry. The project aims to develop support for practitioners to adopt and benefit from CSE principles.

As part of the project, we conduct a multiple case study [18]. The studied cases are established organizations with mature products already in the market. The unit of analysis is the current software delivery process in developing market-driven software-intensive products. Our primary data collection methods are interviews, workshops, and seminars. Our analysis focused on the process, identification of bottlenecks, and opportunities to improve the efficiency and effectiveness of the software delivery process.

This paper is based on results from three partner organizations. We name companies A, B, and C to maintain their anonymity. The work with these organizations was conducted between January 2022 - February 2023.

Company A is a large mobile telecommunications hardware and software provider in Sweden. We were involved with a part of the organization developing business-critical software systems for mobile telecommunications operators. Their software release process is based on a quarterly release cycle. To achieve exceptional quality and compliance, the release process is concluded by a sign-off stage involving many stakeholders representing trade compliance, security, business, engineering, and customer representatives, among others.

Company B is an audio streaming services company based in Sweden. Their offerings include mobile apps, services, and tools to bring musicians, audio content creators, and listeners together. Their organization is relatively new and characterized by a flat structure, agility, and flexibility. Teams can release software changes within hours. However, coordinating large and cross-cutting efforts is challenging.

Company C is a mobile telecommunications operator in Scandinavia. Their offerings consist of both consumer and business oriented mobile telecommunications services. The organization also develops tools for internal business support. Internal efficiency and speed of getting software changes from an idea to production are their major concerns.

4 Results

The first interview rounds focused on understanding how software development and delivery process works and what known bottlenecks are. Later interview and workshop rounds were focused on specific bottlenecks.

The interviews quickly revealed that all three organizations had already implemented some degree of automation and attempted to streamline their engineering processes. For instance, test, build, and integration automation was successfully adopted and led to substantial perceived improvements. However, the automation of development tasks is isolated in the development and testing stage of the process, see Fig. 1.

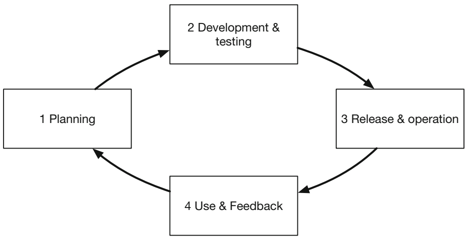

Fig. 1. Overview of the key steps of CSE

Further interviews with companies A and C revealed that significant bottlenecks to rapidly delivering changes occur after development is considered completed. That is, preparing and handing over completed software for release and operations, see Fig. 1. Namely, the handover includes preparing documentation to ensure knowledge transfer, seeking sign-offs from multiple stakeholders, and verifying regulatory compliance, among other activities.

All our partner companies mentioned the intention to use product telemetry and customer feedback to inform further product planning, see Steps 1 and 4 in Fig. 1. However, access to customer environments and data can be problematic due to security and privacy concerns, service level agreements, business risks, lack of trust, or inadequate tooling to manage large volumes of data. Furthermore, due to organizational silos and the lack of transparency, the existing data may not be available to other stakeholders.

Further interviews and workshops revealed that the main obstacles to smooth and continuous software delivery are misinterpreting organizational objectives, leading to organizational silos and stakeholders protecting their interests. This inter-team conflict manifested largely as issues with processes, and teams had difficulty overcoming them. Conflicts like this are not unique to CSE. However, some specific issues arise in understanding and addressing them that need to be studied so that future organizations can address them more easily. It is possible to address these conflicts through analysis and mitigation efforts to improve the performance of CSE efforts.

Further insights from work with companies led us to identify catalysts, symptoms, and outcomes associated with adopting continuous practices and unresolved organizational conflicts.

4.1 Catalysts of Organizational Conflict Hindering the Adoption of CSE Principles

Catalysts do not necessarily lead to conflict or other issues, however, they are mentioned as a context where issues are more likely to emerge.

Catalyst-I: Functional organizational structures emerge from how the organization is built. Our interviewees pointed out that companies are often structured around R&D, operations, sales, product planning, strategic management, and customer support functions. In such organizations, each unit has a distinct function and objectives to fulfill. Any improvements are often limited to each function without considering a broader picture. A consequence of such structures and local optimizations are organizational silos [19].

Companies A and C pointed out that a cross-cutting end-to-end software delivery, see Fig. 1, requires the collaboration of multiple organizational functions. Communication between the silos is often tricky due to a lack of context, mismatching goals, and different specializations. Slow release cycles contribute to lost knowledge and context. Due to issues like artificial boundaries, unspecified goal alignment, or lack of oversight into potential issues, there is an increased risk of conflict between the teams. As one interviewee from *Company A* put it:

> *"R&D teams have no idea of how their work is integrated and delivered to customers. We often discover many issues late in the process and we have to go back [to development] or try to find solutions on-the-fly."*

In Company B, the primary organizational unit is a cross-functional team. Each team comprises a product manager, engineers, infrastructure and delivery specialists, data scientists, and other roles. Such a structure ensures that the complete planning, development, delivery, and feedback collection cycle, see Fig. 1, can be executed within one team. This structure keeps communication distances small and minimizes cross-unit communication and coordination.

Catalyst-II: Inimical processes lead to sub-optimal organizational performance due to lack of flexibility. We observe that organizations have a tendency to grow their processes by adding more steps and including new stakeholders. Processes become inflated due to over-complicated procedures with many stakeholders who do not necessarily understand their role in the process or share the same goals. At the same time, our industry partners report the need to become faster and offer new types of software deliveries. Our interviewees shared their interest in simplifying processes by removing activities and stakeholders. As one interviewee put it:

> *"Our [inimical] process did not help us to catch the log4j issue. It only delayed us in releasing the fix."*

Company-A pointed out that a process built for steady delivery of business-critical software hinders rapid delivery of security updates or experimental features. When the process and the involved stakeholders do not permit enough room for flexibility, it leads to frustration and sub-optimal performance. Moreover, frustration leads stakeholders to seek shortcuts and use their influence in the organization to get things done. Importantly, processes involving ad-hoc negotiations between stakeholders are difficult to automate and streamline.

Catalyst-III: Gatekeeping managers have a role in halting a process until certain conditions are met. They are often part of inimical processes (Catalyst-II). For instance, a gatekeeper would be a senior manager signing off a release for delivery to customers, a council scrutinizing reports to negotiate if the software meets a quality threshold, and alike. While some level of gatekeeping is necessary, a challenge arises when the gatekeeper lacks contextual knowledge and in-depth understanding of the artifacts produced by the process.

As reported by Company A, the gatekeepers are often responsible for signing-off many different artifacts. They have a limited capacity to have an in-depth understanding of what they are signing off on. Getting the gatekeeper's knowledge to a sufficient level for an informed decision takes time and is impractical, given the size and complexity of the software. Slow release cycles and parallel processes lead to the lost context that needs refreshing every time. Hence, the gatekeeper is unlikely to fulfill its function and hinder the pace of software deliveries. As one interviewee put it:

> *"To get an approval, I need to book a meeting with 20 stakeholders. I send them information upfront, but they do not read it in many cases. This repeats every six months"*

In Company B, the gatekeepers are primarily within the team responsible for conducting the work. Thus, they have full contextual knowledge and understanding of the artifacts to make an informed decision.

Based on company comments, it did not appear that the experiences associated with gatekeeping were amicably resolved. Instead, process conflict defined the interaction, where no side could agree on particular operations. Team members who disagree with a set process are likely to become frustrated, and this conflict can impact overall performance and lead to further conflicts between people.

Catalyst-IV: Lack of Autonomy and Support for Conflict Resolution. The above catalysts are exacerbated by the lack of autonomy and support for resolving conflicts. Companies A, B, and C follow agile software engineering practices and empower their engineers to make decisions to a certain degree. However, we observe that organizational conflicts are likely to occur during the handover of artifacts from one organizational unit to another.

Different organizational units may follow different principles and fall under different managers. Thus, resolving any disputes may require an agreement between the parties that they have the authority to resolve the conflict or requires taking several steps up the management chain. As illustrated by Company A, different organizational units often have different objectives and are physically located on different continents. The negotiations between organizational units happen, however, they are repeated at every software delivery cycle, causing delays and frustration. The involved parties need the authority to adjust their ways of working. Companies B and C are smaller, and engineers have more authority to adjust their ways of working. As *Company B* put it:

"We have the authority and the responsibility to do the right thing."

In particular, team members, and project managers need to be adequately empowered and trained to resolve interpersonal conflict. There will be negative impacts. In CSE efforts, the team manager may be able to negotiate over particular deliverables and tasks, but issues with processes are less likely to be managed effectively. Additionally, process conflicts are more likely to propagate to other areas than lower-level task issues, leading to broader negative impacts.

4.2 Symptoms of Underlying Conflicts

Process analysis in Companies A, B, and C revealed a number of challenges hindering rapid software delivery. In our study, these challenges emerged as symptoms of deeper challenges.

Notably, the adverse effects emerge as an overhead for every software delivery cycle. Increasing the pace of software deliveries without optimizing the overhead would compound the overhead and waste.

In our study, we identify the following symptoms of underlying organizational conflicts:

Symptom-A: Repeated Negotiation Every Time an Activity or Process is Performed. All partners mentioned that they depend on meetings to coordinate work. However, interviewees from Company A described that they often have meetings to renegotiate certain decisions every time a process is performed. Such meetings take calendar time to schedule and require informing the stakeholders about the decision context, and in many cases, add little value because the outcome of the decision is expected to stay the same. Our interviewees mentioned that they are used to a release sign-off meeting with about 20 stakeholders formally approving a release. Such a process is acceptable with quarterly releases, however, is impractical when the organization aims to achieve faster release cycles.

Due to inimical processes (Catalyst-III) and the lack of autonomy (Catalyst-IV), stakeholders cannot skip unnecessary negotiations. An individual project or team member does not have the authority to change the process. Should they attempt to do so, more conflict could erupt. In this scenario, the question becomes "who is responsible for process improvement?" or "how do we achieve trust in faster release cycles?." A party such as a senior manager or project sponsor must be empowered to arbitrate or oversee the removal of processes that act as impediments for teams to change outdated procedures while maintaining the autonomy of teams.

Symptom-B: Organizational Constraints Hindering Effective Work. The organizational constraints arise from organizational structures, internal policies, lack of transparency, and the lack of efficient mechanisms to adjust the constraints. In its most severe case, organizational constraints manifest as competing silos.

Company B's main organizational unit is a small, autonomous, cross-functional team. Such a structure makes the organization very efficient in delivering work on a team level. However, coordinating the work of many autonomous teams to deliver more extensive work is challenging. The challenge arises from juggling different priorities on different levels. For instance, a team may be required to support multiple other teams while delivering on their objectives.

Company A has a functional structure, and organizational units have different objectives and specializations. For instance, the responsibility to deliver working software to customers is delegated to one final integration team. Other teams provide software components and have a partial picture of how their components will be integrated and used. Thus, they cannot make informed decisions about their work.

A shared concern in Companies A and B is access to product and process data. The process data is required to enable transparency of other teams' progress and efficient work planning. Product usage data is required to infer customer preferences and steer product development work. However, due to data protection concerns, lack of tooling, and ad-hoc data collection methods, such data is not freely available throughout the organizations.

Company B has an internal team and processes responsible for aggregating data from various sources. However, due to differences in how teams plan their work, there needs to be oversight of the current work progress and resource availability. Company A has an overview of the work progress due to the central planning function, however the data of how software is used is not shared across organizational functions.

Symptom-C: Blocking Task Dependencies. Task dependency arises from the need for multiple organizational units to collaborate and coordinate toward a common goal. Challenges arise when one unit needs to wait until another completes its part. Or when there is a gap between what one unit delivers and what another expects.

In Company A, R&D task dependencies are minimized and well managed. However, difficulties arise in coordination with software delivery, operations, and sales units. Due to the complexity of software, bespoke versions delivered to customers, and lack of transparency, the deliveries do not always match the expectations. Filling the gaps requires substantial effort just before releasing software to customers.

In Company B, teams can collaborate in different constellations. Coordinating and scheduling a large volume of tasks is a major challenge. The challenge can be partly attributed to the lack of company wide task and resources tracking system.

Symptom-D: Utilization of Personal Contacts and Process Shortcuts. Inimical processes (Catalyst-II) and difficulties in reaching consensus push stakeholders to use their influence, networks, and process loopholes to accomplish their objectives.

Several interviewees mentioned using personal relationships to reach out to other stakeholders and influence them to behave in a certain way. We observe that such behavior is more prevalent among more senior employees. At the same time, more junior interviewees mentioned the difficulties of attaining the same outcome through formal channels.

Another manifestation of this symptom is labeling artifacts in a certain way to simplify the process. For instance, an interviewee mentioned labeling a major update as a hot fix to avoid a lengthy, and in their view, pointless review process.

4.3 Outcomes from Unresolved Organizational Conflicts

Our partner companies report a number of negative outcomes associated with the symptoms. Importantly, the outcomes become significant concerns when organizations plan to speed up their release cycles and improve internal efficiency and effectiveness.

Outcome-A: Wasted time. Leaping organizational boundaries, renegotiating the same decisions, involving distant stakeholders in making critical decisions takes time away from more value creating activities. The time is wasted every time a process runs.

Outcome-B: Lack of flexibility. Inimical processes, dependency of personal networks to get things done, and lack of autonomy to resolve inefficiencies limits the organizational ability to adapt to new market and business conditions.

Outcome-C: Frustration. Awareness of organizational inefficiencies lead to frustration among stakeholders. In turn, frustration leads to reduced motivation, increased turnover rate, and overall reduced organizational performance.

5 Discussion and Analysis

This research collected information broadly about adopting CSE in large companies with multiple software products. The focus of the data collection was not explicitly aimed at the conflict. Nevertheless, the insights touched upon conflicts throughout the process and bore further investigation.

Our results show that adopting CSE is as much an organizational as it is an engineering challenge. Thus, a study into CSE adoption must balance the technical and social lines of inquiry.

Studying the organizational impediments to CSE adoption falls squarely in the realm of the study of people and organizations, which managers must be familiar with to be successful. The company's experiences in these cases revealed conflicts that organizations are likely to face when implementing CSE and Agile methodologies, more broadly.

5.1 RQ1: What are Symptoms of Organizational Conflict in the Context of Adopting CSE?

There are a number of organizational concerns to be addressed when adopting CSE practices. We separate these concerns into catalysts and symptoms.

The catalysts are not problematic or cause conflict per se. However, our results suggest that functional organizational structures (Catalyst-I), inimical, die-hard processes (Catalyst-II), the culture of sign-offs (Catalyst-III), and the lack of autonomy and support for conflict resolution (Catalyst-IV) create conditions for conflicts to emerge.

Symptoms are signs of deeper issues. We identify that the culture of repeated meetings and decisions with the same outcome (Symptom-A), organizational "red tape" (Symptom-B), blocking task dependencies (Symptom-C), and the utilization of process shortcuts (Symptom-D) signal a deeper conflict.

These results are well aligned with the state-of-the-art in CSE. For instance, cross-functional teams, empowering engineers to decide on their ways of working, and organizational transparency are prerequisites for the successful adoption of CSE [7]. At the same time, the existing literature does not provide a deeper insight on the organizational change management perspective [10,15]

5.2 RQ2: What are the Root Causes of Identified Symptoms from an Organizational Conflict Perspective?

State-of-the-art management practices recognize several groups of factors contributing to organizational conflict. These categories include issues caused by task interdependence, goal incompatibility, ambiguous rules, differences in values and beliefs, resource scarcity, and ineffective communication [15].

We use the causes of organizational conflict to analyze insights from our study. Under each cause for the organizational conflict, we list sub-causes mentioned by our interviewees, see Fig. 2.

Our observations match well with the state-of-the-art in organizational conflicts. However, conflict management has attracted little attention from CSE community. Moreover, discussions with our partners reveal that the adoption of CSE is treated mostly as an engineering problem with the focus on technical aspects. Our findings demonstrate how including conflict management in the CSE adoption strategy could alleviate some of the adoption challenges [16].

Software engineering and delivery of software-intensive products and services are inherently collaborative. Thus, some degree of dependency and conflict is inevitable, see C1.1 in Fig. 2. In Company A, cross-functional dependencies are more common. However, in Company B, the team dependencies are more common, thus conflicts occur on different levels.

Our interviews confirm that collaboration across organizational units indeed causes friction, see C1.2 in Fig. 2. From Company A, we learn that well-coordinated work in the R&D unit can solve many issues. At the same time, the coordination does not extend beyond the functional R&D unit. This is a potential cause of conflict.

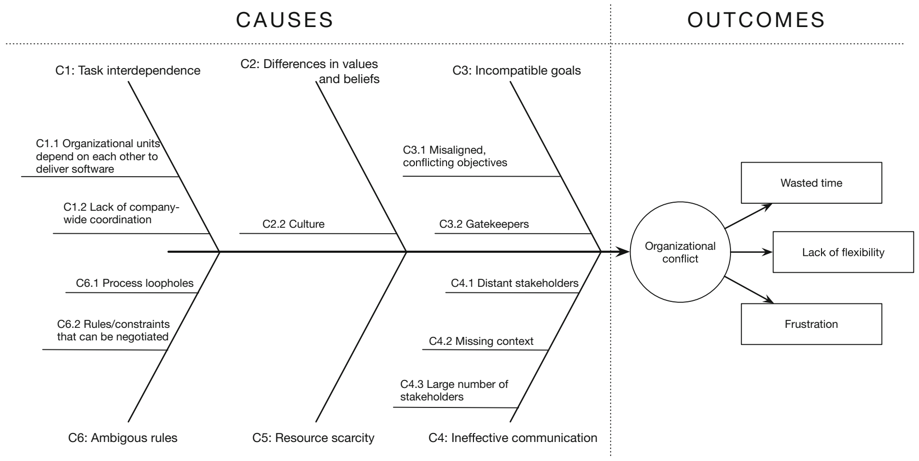
CAUSES
OUTCOMES
C1: Task interdependence
C2: Differences in values and beliefs
C3: Incompatible goals
C1.1 Organizational units depend on each other to deliver software
C1.2 Lack of company-wide coordination
C2.2 Culture
C3.1 Misaligned, conflicting objectives
C3.2 Gatekeepers
C6.1 Process loopholes
C6.2 Rules/constraints that can be negotiated
C4.1 Distant stakeholders
C4.2 Missing context
C4.3 Large number of stakeholders
C6: Ambigous rules
C5: Resource scarcity
C4: Ineffective communication
Organizational conflict
Wasted time
Lack of flexibility
Frustration

Fig. 2. Root-cause analysis of organizational conflicts in adopting continuous software engineering.

Topics around poorly understood and misaligned organizational objectives were discussed during our interviews, see C3.1, in Fig. 2. We conclude from the interviews that setting multiple organization-wide goals with shared and clear key performance indicators is challenging.

In Company A, the primary goal for a long time had been the quality of services. Consequently, there is a rigorous quality assurance process. Recently, the organization sought to increase delivery speed. This move requires rethinking and streamlining the quality process. In particular, the gatekeepers, see C3.2., need to be replaced with an automated and scalable solution. At the same time, the priorities of quality and speed are not shared, and some fears removing the established process may lead to deteriorating product quality.

In Company B, the primary goal is delivery speed. The organization had been optimized to deliver features to the market fast. At the same time, the ability to track the market relevance of high-level updates is limited.

We observe that many stakeholders across organizational units cannot maintain the full context of the situation, coordinate and make efficient decisions. Consequently, fault lines appear, and organizational conflicts emerge.

This challenge is most prevalent in Company A, where the key stakeholders have a many-to-many mapping with the ongoing projects with long release cycles. Thus, they cannot fully understand each project's context.

The causes and effects of the conflicts we identified are directly related to the shift to CSE. These issues may not arise in other business transformations and require special attention in this context. How these conflicts manifest for different companies differs and depends on the legacy structures and processes, as well as overarching culture and goals. Team members can be conscious of these prospective issues and flag them should they appear. Managers can also be prepared to address them based on best practices in order to maintain performance and meet goals.

Initially, managers can take stock of the catalysts we identified here. These represent the corporate structures that can lead to conflicts later on. Where possible, leaders should prepare for CSE adoption by restructuring some of the organization, or its processes at least, to address these root causes. At a minimum, engineering teams should be prepared to deal with the symptoms that arise from these catalysts. Failing that, (for example, if you read this research after starting a CSE change) team members should be aware of the symptoms that are driven by the catalysts. If these symptoms arise, they can likely be traced back to their root causes at the organizational level to seek solutions. Performing a root cause analysis on symptoms not defined here could also reveal new catalysts.

5.3 RQ3: What are the Advantages of Studying Conflicts in CSE Empirically?

Studying the conflicts in CSE, including their symptoms and catalysts helps prevent them, and also helps too maximize their positive outcomes when conflicts inevitably occur. For instance, since some degree conflict in teams is healthy [8], managers should reflect on them with their teams during reviews and retrospectives, to celebrate the wins and encourage a positive discord. Embracing positive conflict could be a driver for incremental organizational change.

Tracking the frequency of conflicts is also valuable. A lack of conflict may indicate instability in a team or organization, since stakeholders may avoid raising disputes for fear of causing their group to crumble. Instead, observing a manageable amount of healthy conflicts can help indicate team cohesiveness [4]. Managers should clue-in to task conflict that results in positive outcomes, and raise a red flag when teams grow silent.

Lastly, the goals of adopting CSE are meant to close the gaps between the business need for a software solution and its development, without negative impacts to quality in the short- or long-term [7]. As with any Agile transformation, and for any project manager, removing obstacles is the key priority. For CSE, those obstacles have been shown to surface as issues between people more than from technical limitations. Just like change is inevitable, so too is conflict. Organizations should thus prioritize management, and to some extent welcome, disputes that arise during adoption of CSE, particularly between heterogeneous organizational units. This means, for example, facilitating dialogue to focus disparate interests towards shared goals.

By systematically identifying, addressing conflicts that naturally occur, organizations can help ensure the success of a continuous pipeline of software that meets the needs of customers, engineers, operations, and sales.

6 Conclusions and Further Work

In this paper, we report preliminary results from an ongoing study on adopting CSE in three companies. Through interviews and workshops, we observed challenges faced by different teams within each organization. The data regularly included participant discussion of organizational conflicts, defined by differences in the understanding of goals, needs, and interests between teams. Based on this empirical information, we have identified catalysts, symptoms, and outcomes associated with improperly managed organizational conflict that inhibits the adoption of CSE. Unless it is addressed, organizational conflict can hinder the implementation of a continuous software delivery pipeline and the surrounding automation.

We conclude in this early work that analyzing organizational conflict, and managing it constructively, are important parts of streamlining an organization and realizing the benefits of CSE. We propose to recognize *continuous conflict management* as an essential and cross-cutting activity in the toolbox of implementing continuous integration and continuous delivery.

Further work should look to these underlying conflicts, their causes, and their outcomes, as a starting point to identify further conflicts that may arise. Quantitative studies could also be used to measure the frequency of such conflicts and evaluate their impacts.

References

1. Afzalur Rahim, M.: Toward a theory of managing organizational conflict. Int. J. Conflict Manag. **13**(3), 206–235 (2002)
2. Chen, L.: Continuous delivery: overcoming adoption challenges. J. Syst. Softw. **128**, 72–86 (2017)
3. Claps, G.G., Svensson, R.B., Aurum, A.: On the journey to continuous deployment: technical and social challenges along the way. Inf. Softw. Technol. **57**, 21–31 (2015)
4. Coser, L.A.: The Functions of Social Conflict, vol. 9. Routledge (1998)
5. Debbiche, A., Dienér, M., Berntsson Svensson, R.: Challenges when adopting continuous integration: a case study. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) PROFES 2014. LNCS, vol. 8892, pp. 17–32. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13835-0_2
6. Fitzgerald, B., Stol, K.-J.: Continuous software engineering and beyond: trends and challenges. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, pp. 1–9 (2014)
7. Humble, J., Kim, G.: Accelerate: the science of lean software and devops: building and scaling high performing technology organizations. IT Revolution (2018)
8. Jehn, K.A.: A qualitative analysis of conflict types and dimensions in organizational groups. Adm. Sci. Q. 530–557 (1997)
9. Karn, J.S., Cowling, A.J.: Measuring the effect of conflict on software engineering teams. Behav. Res. Methods **40**, 582–589 (2008)
10. Klotins, E., Gorschek, T.: Continuous software engineering in the wild. In: Mendez, D., Wimmer, M., Winkler, D., Biffl, S., Bergsmann, J. (eds.) SWQD 2022. LNBIP, vol. 439, pp. 3–12. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-04115-0_1

11. Klotins, E., Gorschek, T., Sundelin, K., Falk, E.: Towards cost-benefit evaluation for continuous software engineering activities. Empir. Softw. Eng. **27**(6), 157 (2022)
12. Korsgaard, M.A., Soyoung Jeong, S., Mahony, D.M., Pitariu, A.H.: A multilevel view of intragroup conflict. J. Manag. **34**(6), 1222–1252 (2008)
13. Laukkanen, E., Itkonen, J., Lassenius, C.: Problems, causes and solutions when adopting continuous delivery-a systematic literature review. Inf. Softw. Technol. **82**, 55–79 (2017)
14. Laukkanen, E., Paasivaara, M., Arvonen, T.: Stakeholder perceptions of the adoption of continuous integration-a case study. In: 2015 Agile Conference, pp. 11–20. IEEE (2015)
15. Mitchell, D.E.: Causes of Organizational Conflict. Springer, Cham (2017)
16. Neely, S., Stolt, S.: Continuous delivery? easy! just change everything (well, maybe it is not that easy). In: 2013 Agile Conference, pp. 121–128. IEEE (2013)
17. Power, K.: Impediment impact diagrams: understanding the impact of impediments in agile teams and organizations. In: 2014 Agile Conference, pp. 41–51. IEEE (2014)
18. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009)
19. Serrat, O.: Bridging organizational silos. In: Knowledge Solutions: Tools, Methods, and Approaches to Drive Organizational Performance, pp. 711–716 (2017)
20. Siddique, L., Hussein, B.A.: Grounded theory study of conflicts in Norwegian agile software projects: the project managers' perspective. J. Eng. Project Prod. Manag. **2**, 120–135 (2016)
21. Williams, R.A.: Conflict propagation within large technology and software engineering programmes: a multi-partner enterprise system implementation as case study. IEEE Access **7**, 167696–167713 (2019)
22. Zhang, X., Stafford, T.F., Hu, T., Dai, H.: Measuring task conflict and person conflict in software testing. ACM Trans. Softw. Eng. Methodol. (TOSEM) **29**(4), 1–19 (2020)