A Case Study of Continuous Adoption in the Norwegian Public Sector

Astri Barbala
SINTEF
Trondheim, Norway
astri.barbala@sintef.no

Tor Sporsem
SINTEF
Trondheim, Norway
tor.sporsem@sintef.no

Klaas-Jan Stol
University College Cork, Lero, SINTEF
Cork, Ireland
klaas-jan.stol@sintef.no

Abstract

The Norwegian public sector has become increasingly software-intensive. To enable faster software delivery involving frequent deployments, software development teams building new solutions for public sector employees and citizens have started to embrace a Continuous Software Engineering (CSE) approach. Research on CSE has primarily had an inward focus on the development practices including Continuous Integration and Delivery. Far fewer studies have had an outward focus that considers the involvement of users of a system that is delivered incrementally, and 'continuously.' Further, most CSE research is conducted in commercial settings, where the quest to innovate and retain users in a competitive market is key. This paper presents a case study of new systems development in the Norwegian public sector. Our analysis identified the concept of "Continuous Adoption" as a distinct concept from Continuous Use. This paper extends the CSE literature, and complements traditional adoption literature. This paper presents a definition and identifies three key dimensions, namely transparency, feedback, and evolving context, and illustrates these using an in-depth analysis of a longitudinal case study.

Keywords: Continuous Software Engineering, Public Sector, Continuous Adoption.

1. Introduction

Digital transformation efforts have moved center-stage on the agenda of politicians, public administration stakeholders, software developers, and research communities over the last decade and are radically changing how public sector organizations are organized and operated (Plesner et al., 2018). The Norwegian public sector is undergoing a digital transformation (Moe & Mikalsen, 2020) that aims to increase data-sharing between the various stakeholders in public administration and create more effective and user-friendly services. The Norwegian public sector has become increasingly software intensive, which means that both employees in public administration and Norwegian citizens must maneuver these digital solutions. A major trend in the software industry is Continuous Software Engineering (CSE). CSE captures a wide range of activities including Continuous Planning, Continuous Integration, but also Continuous Use; Fitzgerald and Stol (2017) have labeled this 'Continuous *'. Their Continuous * framework suggests that besides long-established practices such as continuous integration, software companies could also benefit from focusing on other continuous practices such as Continuous Use, which refers to the continued use of software systems after the *initial* adoption (Fitzgerald & Stol, 2017). However, this concept, which focuses on the relationship between development teams and user groups, has remained largely untheorized to date. Research on CSE practices has first and foremost focused *inward* on the development process, studying e.g., experimentation (Auer & Felderer, 2018; Ros et al., 2022; Yaman et al., 2020) and cost-benefit evaluations (Klotins et al., 2022) in a continuous development process. Very few studies have focused *outward* on the user side in the context of CSE, and user-focused studies have mainly looked at how development teams use feedback and integrate it into their continued work (e.g., Fabijan et al., 2017; Maalej et al., 2009).

An outward view on CSE involves the adoption of the software that is developed and includes attention for concerns of users. While there is considerable literature on adoption, prior frameworks and theories do not fit well with the CSE approach, which Bosch et al. (2018) have labeled a 'paradigm shift.' In this paper, we coin the term "Continuous Adoption" to refer to the continuous adoption practices that involve both software development (supply side) and software use (user side). We define Continuous Adoption as a distinct concept from Continuous Use and initial adoption, and identify three integral features that characterize the concept which are central for determining its success, namely transparency, feedback, and evolving context.

URI: https://hdl.handle.net/10125/106626
978-0-9981331-7-1

Given the nascent state of research on software adoption in a continuous development context, we conducted a longitudinal case study at NAV, a large public organization in Norway responsible for social welfare and labor services. The expert users in this case study are caseworkers in state aid centers who evaluate and process digital applications for various auxiliary aids like wheelchairs and hearing aid devices. With several thousand employees and servicing large parts of the Norwegian population at different life stages, NAV is a prime example of a large, complex organization. In recent years, NAV has moved away from being dependent on large consultancy companies, to in-house software product teams organized by specific domain. The organization comprises several divisions, with many interdependencies among them. The development teams have moved towards pursuing a CSE approach, with an aim to deliver and deploy software frequently. The frequency of deployment amongst NAV teams has increased from a few releases per year to several a day.

Our study goal was to develop a better understanding of the continuous adoption process that takes place in a CSE context, from the perspective of both developers and users. As the focus here is on the process and not the result—which, in CSE is a matter of never-ending negotiations—we employ a process perspective to the study, which allows for elaborating on *"underlying dynamics [revealing] how and why outcomes are reached over time"* (Fitzgerald, 2009). We ask:

RQ: *How is the Continuous Software Engineering development process experienced by the supply and user sides in a public sector, and which challenges do expert users experience in adoption of the software?*

Continuous Adoption takes place at the intersection of software development and software use, and thus it is necessary to understand both perspectives to provide a deep understanding of the adoption process in the Continuous * framework. We interviewed and observed both caseworkers and the developers of digital solutions.

The paper is organized as follows. Section 2 presents related work and the conceptual foundation of our study, followed by a presentation of our methodological approach and data in Sec. 3. We then present our results in Sec. 4. Section 5 discusses the main findings and limitations, and directions for future work.

2. Background and Related Work

2.1. Continuous Software Engineering

Mirroring the term 'DevOps' (Debois, 2011), Fitzgerald and Stol (2017) coined the term 'BizDev' to argue for closer integration between business strategy and development. In their discussion of CSE, Fitzgerald and Stol make a link to the concept of 'flow' which originated in Lean manufacturing and encoded in "Lean Thinking" (Womack & Jones, 2003, p.180), and which refers to a *"connected set of value-creating actions."* They argue that Lean Thinking's explicit focus on an end-to-end process may be better suited for a 'continuous' approach than agile software methods, which focus less on a holistic chain of integrated practices. Although they identified several significant challenges for the practice of CSE, including a potentially misplaced focus on speed, Fitzgerald and Stol concluded that the need for more holistic and integrated approaches within software engineering demands clear and rigorous definitions. CSE comprises several related but distinct practices, but three of these in particular are applicable to discussing software adoption in a continuous context, namely Continuous Use, Continuous Trust, and Continuous Innovation.

Continuous Use considers users' continued employment of a product after its initial adoption. Literature on Continuous Use places a focus on how the user as existing costumer chooses to keep using the product (Fitzgerald & Stol, 2017), and how this continued use is vital for a vendor to receive economic payoff. Again, the focus is mainly inward and on how development teams can work in order to facilitate for this use and its accompanying financial reward. Fitzgerald and Stol (2017) further point out that *"many studies consider intention to continue using a system rather than the actual continuous use,"* meaning that very few studies report detailed accounts of user experiences in the context of CSE. At the same time, the term Continuous Use lacks attention for the broader context of deploying the software into actual practice, and the associated negotiations, appropriation, and habits that happen when software is adopted.

Closely related, intertwined even, to Continuous Use is the concept Continuous Trust, where trust is seen as a prerequisite for users to keep using a product in question. Fitzgerald and Stol (2017) defined Continuous Trust as: *"trust developed over time as a result of interactions based on the belief that a vendor will act cooperatively to fulfill customer expectations without exploiting their vulnerabilities."* Hence, the term implies a wider understanding than merely trusting the software itself; it also implies the reliability and reputation of the vendor or organization. Trust is regarded one of the central human values increasingly seen as critical in all stages of software development, although the studies incorporating a focus on trust have mainly looked at e-commerce, freelancing, and crowd-sourcing platforms (Hussain et al., 2022). In discussing how establishing trust with customers is intertwined with the

implementation of user feedback, Klotins et al. (2022) wrote: *"The primary benefits from continuous use arise from closer, longitudinal relationships with customers and end-users. The closer relationship enables more opportunities for feedback collection, builds trust, and improves overall customer satisfaction."*

Similar to the conceptualization of Continuous Use, then, Continuous Trust uses terms such as "vendor" and "customer" in its definition. While this does not necessarily exclude public sector IT systems, the main concern expressed here is that of vendors seeking to retain a customer base by acting in a way that customers would expect. Indeed, to the best of our knowledge, continuous trust has only been studied in the context of commercial operators (Gefen et al., 2003; Hoehle & Huff, 2012). Whereas in the private sector users of IT services may have several alternatives, this is not the case in the public sector. IT systems in the public sector have the exclusive mandate from a country's government to provide services to its citizens. Citizens have no choice but to use these systems to avail of the services that a government offers. Another relevant study within the CSE literature that underline the importance of trust-building is Mattos et al. (2020)'s investigation of how high levels of trust and close collaboration with customers were vital for successful Continuous Experimentation (CE). They underlined that a key to building a trust relationship is transparency between the user and the company, where: *"the customer needs to understand the need for the deployment, have a clear vision of how it can impact the system and what are the potential benefits."*

The third term that relates to our suggested concept of Continuous Adoption is Continuous Innovation. This concerns how *"new ideas are transformed to create business value for customers"* (Fitzgerald & Stol, 2017) with an emphasis on the planning process, and although the focus is yet again largely on financial opportunities (e.g., Wiedemann et al. (2019)), the concept is also useful for the public sector; in our case the aim is to develop novel ideas to use data and simplify case processing.

Due to the implications brought about by CSE, Bosch et al. (2018) refer to the current changes in software development as a "paradigm shift," where software-intensive organizations are seeing a need to complement traditional requirement-driven approaches with data-driven practices and use of machine learning and artificial intelligence (AI). We argue that this paradigm shift also entails a change regarding how users are integrated into the development process, which we seek to elaborate in this study.

2.2. Technology Adoption

Several conceptual models have been proposed to help understand how technology is integrated into users' lives. Although we contend that existing frameworks to discuss the findings in our study do not suffice, the Continuous Adoption concept draws from both the CSE literature and technology adoption literature within the Information Systems (IS) and Science and Technology Studies fields to pinpoint the aspects at stake in the context of public sector caseworkers integrating new digital solutions.

Among the best-known frameworks is the Technology Acceptance Model (TAM) (Davis, 1989) which posits that two factors determine whether a computer system will be accepted by its potential users: 1) perceived usefulness, and 2) perceived ease of use. However, this model focuses on the *initial* adoption of technology, and does not pay attention to the supply side and the development processes, nor does it consider the continuous nature of integrating versions of a system inot a user context. TAM has had many extensions since its initial introduction, among the most central ones being the Unified Theory of Acceptance and Use of Technology (UTAUT), which regards users' intentions of using specific technology (Venkatesh et al., 2003). Similarly, the domestication framework (Berker et al., 2005) underlines how different uses of technology are negotiated into people's lives, *"influenced by choice as well as discipline, by enthusiasm as well as resistance"* (Sørensen, 2005). This model primarily focuses on individual users and the meaning-making tactics utilized by them to 'tame' technological devices or systems to fit their existing everyday habits. Although useful for this study for its focus on the continuous negotiations that take place when integrating software into everyday life contexts, the domestication framework also lacks attention to how technology development affects the negotiation process.

A growing scholarship is investigating technology adoption taking place within the public sector, however usually with a focus on challenges and implications taking place at the initial implementation on new technology. Reddick (2009)'s study of American local governments' adoption of centralized customer service systems is a much-cited example. It points to how organizational, environmental, technological, and geographical factors are central for the success of its implementation. This is, however, a statistical study and focuses on initial adoption, and hence differs from our research. Similar studies are conducted in other countries, including Kaliannan et al. (2007)'s insights into how the Malaysian government implemented

e-government solutions and Takahashi et al. (2020)'s study of digital farm managing systems in sub-Saharan Africa. Again, the focus is on initial implementation and on organizational and financial implications rather than on users' and suppliers' experiences with a continued adoption process. By adopting the CSE framework and focusing on qualitative data, we contribute novel insights to the literature on technology adoption in public sector contexts.

3. Methodology

We conducted an exploratory case study (Runeson & Höst, 2009), drawing on observations and interview data from a public sector development team and caseworkers at four state aid centers that use the software built by the team. Case study research can be a fruitful approach to develop novel theoretical concepts by observing actual practices (Meredith, 1998). Additionally, we drew on strategy documents and reports on the status quo on digitalization efforts within the organization. Both traditional and digital observations were conducted, entailing observations of both the NAV development team and selected caseworkers. Both groups were observed in their everyday working environment. To ensure a detailed overview and representation, the aid centers are placed in different parts of Norway and comprise both centers in large and small municipalities. The users we studied can be described as expert users, and the software in question is tailored to their needs in their everyday work contexts.

3.1. Background to the Case

We studied the development of a new digital case management system at NAV. The system is tailored to caseworkers in 12 aid centers located in municipalities across Norway, which serve citizens in their local regions. Between 100 and 200 caseworkers use the software in question in their everyday work, yet the different caseworker units focus on different technological aid areas and hence do not process the same digital applications. Aid Center 4 and the development team jointly designed a new part of the system to satisfy new public policy requirements. This regards one specific type of technological aid that now requires the public to choose between different applications when applying for those. As our study shows, the intertwining of new technology and new public policies further complicates the continuous software adoption process, and we here contribute to the literature pinpointing the entanglements of technology and law (Cordella & Gualdi, 2019; Gualdi & Cordella, 2022).

We refer to the development team we studied as Team Welfare. Their team structures employ principles for typical agile teams: Product Owner, developers, designers, and other domain-specific roles.

3.2. Data Collection and Analysis

We collected data through interviews and observations (see Table 1). The interviews were semi-structured and lasted between 30 and 60 minutes. In total, we conducted 18 interviews with 19 informants (the development team Product Owner was interviewed both at the beginning and the end of the data collection period, and three informants in Aid Center 4 were interviewed together). Seven of the interviews in the aid centers can be described as field interviews, entailing that these were more informal conversations during our observations. These field interviews lasted more than 30 minutes each; in addition, we also had many, shorter and more casual conversations. The interviews focused on the informants' attitudes toward the software, the development process, and how this was experienced both by developers and users.

We conducted approximately 30 hours of observations at aid centers (ca. 7 hours in each) and 4 hours of observations of the development team. Additionally, we attended two online meetings and analyzed conversations in the chat rooms within NAV's online meeting system. The field notes comprise approximately 25 pages of written text, noting in the aid centers how the caseworkers made use of the different software during the day, how they spoke about both the software in question and their attitudes towards the development team, and the workaround tactics they had developed.

Interviews were recorded, transcribed, and coded open-ended, and we used NVivo to identify and structure common codes into higher level concepts. The Constant Comparison Method (Seaman, 1999) used for coding and memoing and findings were analyzed using an abductive thematic analysis which focuses on a dialogue between the empirical data material and theory. In order to structure the codes into higher level themes we followed Saldaña (2013)'s pragmatic approach to thematic analysis. As some of the analysis was conducted jointly by the research team, our approach was also informed by the collaborative thematic analysis approach (Eggebø, 2020), entailing that we discussed back and forth before deciding and agreeing on the final themes. Three major themes emerged from the data analysis: transparency, feedback and evolving context. Within these overarching themes, the most important subthemes were techno-legal hurdles, types of communication (e.g., online meeting, instant messenger (IM) messages, face-to-face communication), planning challenges for aid

Table 1. Data collection activities

Period	Activity	Interviewees
Dec 2022-June 2023 NAV	7 interviews + 1 workday observing the development team + attending 1 online meeting + 1 feedback presentation	• Product owner (x2) • Developer 1 • Developer 2 • Designer • Communications advisor • Legal advisor • Department director
Feb 2023 Aid Center 1	2 interviews + 1 workday observation + attending 1 online meeting	• Caseworker 1 • Digital advisor/ caseworker
April 2023 Aid Center 2	5 interviews + 1 workday observation	• Caseworker 1 • Caseworker 2 • Special aid advisor • Center director • Digital advisor
May 2023 Aid Center 3	2 interviews + 1 workday observation	• Caseworker 1 • Caseworker 2
June 2023 Aid Center 4	1 group interview + 1 workday observation	• Caseworker • Special aid advisor • Legal consultant/Team coordinator

centers, guilty conscience among the development team, and lack of interdisciplinary understanding.

4. Findings

Our analysis led to the emergent theme of 'Continuous Adoption' (CA), a continuous process of adopting new versions of software that is actively being developed. We now present a definition (Sec. 4.1), and three key elements of CA (Sections 4.2, 4.3, and 4.4).

4.1. Continuous Adoption

The analysis revealed themes and issues that existing technology adoption frameworks do not account for. Thus, we identified a need to propose a new concept that pinpoints the integration process between development and use, and the continuous negotiations taking place in this respect in order to provide a nuanced discussion of our findings. Recognizing the key limitations of current adoption literature, we coin the term "Continuous Adoption," to extend the continuous software engineering literature, and contrast with traditional notions of *initial* adoption. Specifically, within Fitzgerald and Stol (2017)'s "Continuous *" framework, Continuous Adoption sits closely with Continuous Use and Continuous Trust. We define Continuous Adoption (CA) as:

> "a continuous process of adopting new versions of a software system that involves negotiation between software providers and software users, which takes into consideration concerns of users, such as transparency, ease-of-use, and responsiveness, while also considering the concerns of software providers, such as policy, standards, and regulation."

Similar to other practices of CSE, CA does not have a distinct ending, but rather indicates that the adoption of the software happens continuously and at the same pace as the software is being deployed by the development team building it. Continuous Adoption also underlines the importance of the context of continued use, as users must adopt both the software *and* adjust to potentially new ways of working and new public policies integrated into the software in question.

Lastly and importantly, Continuous Adoption implies that there is a provider side offering something that is to be adopted, which the term "continuous use" lacks. This is another central aspect of the concept of Continuous Adoption underlining the need for a novel concept here; the part continuously providing software for the users to adopt are directly involved in the outcome of the adoption process and the feedback loop. Our thematic analysis resulted in three main themes that we argue make up the key aspects of Continuous Adoption, namely transparency, feedback, and evolving context.

4.2. Transparency

When Team Welfare embarked on the development of their new case processing software, they introduced a paradigm shift in the way caseworkers were included in the development process. Unlike the traditional approach of receiving plans and fixed release dates, users were now presented with a continuous development process where software would be delivered incrementally. This necessitated a change in mindset for caseworkers who had to adapt to the dynamic nature of CSE. They needed to grasp the concept that their own adoption and feedback as users would directly impact the development team's priorities and the timing of new feature releases.

In response to the incremental software releases, caseworkers must now adapt their working methods and organizational practices to incorporate the new software into their daily workflows. When confronted with the continuous way of adopting software, some of the caseworkers expressed a loss of control over their work processes and schedules, as they had to align with the release pace determined by the development team and their prioritization of features. This was linked to a lack of understanding of how the development teams were

organized, how they scheduled deliveries, and, perhaps most importantly, why they worked the way they did. Pinpointing this, a caseworker explained: *"[I]t's hard to be part of this process because it's difficult to know what happens [as the development is] moving forward. Because people come and ask, 'What happens now,' and we know nothing until we receive the news just before [a new function is] released."*

This highlighted the fact that the continuous development process was mostly concealed from the users. Many caseworkers expressed frustration with the developers' work method, and they feared it would make it difficult to plan resources and schedule holidays at the aid centers. Uncertainty surrounding the deployment of new or improved features meant they were never sure when to expect changes.

The developers in Team Welfare acknowledged the caseworkers' struggles in adapting to the continuous development but found it difficult to do something about it. The challenge was that there were no natural points in time to inform about new feature releases or changes made. One developer asked, *"When is a change big enough to invest time in informing the users?"* The development team found it difficult to understand the information needs of caseworkers, given their limited knowledge of caseworkers' work processes. They also did not want to confuse and overwhelm the caseworker users with too much information whenever a new release was ready. However, they underlined that if a new release would come with considerable changes, then users should be notified of this. Finding the right balance, or 'threshold' for reaching out to users was a challenge for the development team. This again points to a lack of communication and mutual understanding about the different work processes from the development and the user sides, something that would take both time and effort to establish. Complicating this, the development team would have to familiarize themselves with completely different aid centers around the country, each with their own distinct culture and geographical features.

Another challenge arose from the difficulty of providing timely information about planned features when there was uncertainty surrounding their actual release. Users saw the need to be informed about upcoming features in order to make necessary preparations. This could involve adjusting staffing levels to accommodate the change or acquiring knowledge about how it would impact their work processes. This situation presented a dilemma: if the information was shared too late, users were unable to adequately prepare, but if it was provided too early, users might prepare for something that ultimately would not materialize. The development team's communication advisor said, *"We could choose to tell users that [a release] might happen next week. But we have also had a pilot solution for two years now because we didn't know exactly what would happen. I don't want to hype anything if it isn't going to be released."*

Informing caseworkers about new features that were ultimately not released would lead to a loss of trust and disappointment. This had a noticeable effect on some caseworkers who became disinterested in adopting the new software.

These findings highlight how the nature of CSE and its elusiveness as perceived by users can erode user trust and hinder successful software adoption. In this case, the limited number of caseworkers, ranging from 110 to 180, made it crucial to prioritize their satisfaction in order to maintain a productive collaboration between them and the developers. The caseworkers indicated that they missed transparency of the development process.

4.3. Feedback

A frequent error message could be observed flashing across the screen repeatedly when case workers used the new software: *"Could not retrieve the case. There has been a technical error. Contact the developers in Team Welfare."* We saw this error message several times during our observation of caseworkers. However, contacting the developers in Team Welfare about this issue appeared to be challenging in practice, because case workers were scattered across Norway, and the only means of communication they had with the product team was through channels in the online meeting system. Caseworkers were reluctant to use these channels to provide feedback. When we asked why, several reasons were revealed.

Several caseworkers shared that they felt scared or intimidated when they had to use channels within their virtual meeting software to report errors or post suggestions for improvement and changes in the software based on what they perceived were missing for them to do their work properly. The reason was their frequent uncertainty about whether the current software was faulty or that they might be using the software incorrectly. They were afraid of embarrassing themselves in front of other members of the channel they did not know when making reports in an open channel, as one caseworker explained, *"You have an audience when you write in a channel."* Some of them chose to directly contact the Product Owner within the development team when they experienced this insecurity. Caseworkers perceived such direct contact as 'safer,' and they were more comfortable asking seemingly stupid questions and ensuring they had understood things correctly. However, we found

that only the caseworkers who had already established a relationship with the Product Owner would do this.

One of the reasons why caseworkers struggled to provide feedback was the difficulty in articulating their thoughts effectively. The field of assistive technology is complex, making it time-consuming to explain to developers why even minor errors or deficiencies can lead to significant problems. One caseworker noted that the developers seemingly were unfamiliar with basic domain knowledge and terminology: *"For instance, [the developers] don't seem to know what an exchange form is,"* making it challenging to explain its purpose and discouraging her from spending time providing feedback. Further, caseworkers felt insecure about using technical jargon commonly used by developers. Without these terms, it became difficult for them to accurately describe errors and suggest improvements. Despite their limited knowledge of software development, all caseworkers we observed displayed humility and respect towards the software developers, acknowledging the complexity and challenges associated with their work towards digitalizing the public sector. However, the caseworkers also felt like a liability or burden when they were not able to articulate themselves accurately, further discouraging them from providing feedback.

Another factor that influenced the feedback from caseworkers was the response given by the development team. Team Welfare continuously prioritized tasks for each development iteration. This meant that the team could not provide users with information about the impact of their feedback, as priorities were constantly changing. Eventually, as tasks were completed, there was no trailing back to the initial feedback and the person who provided it. This made caseworkers uncertain as to whether their effort of providing feedback was worth it, as one caseworker explained: *"There is no motivation in itself to provide feedback when you don't know what has been done with the feedback."* As a result, their trust in the development process diminished for some of the users we spoke to, and they began to see providing feedback as a waste of time.

The team members' awareness that informing users was not prioritized was a constant source of guilty conscience. This seemed closely connected to the lack of motivated users, as the team was aware of their responsibilities in informing users about what happened to the feedback, yet they were not always able to prioritize this over other tasks. Thus, CSE not only puts new responsibilities onto users, but also on the development teams: they are expected to continuously follow up and inform their users about changes in the software, despite oftentimes having no clear agenda of future releases.

In the context of software engineering, the continuous adoption of new software necessitates the provision of feedback to customize the software according to user needs and domain requirements. However, this study highlights some challenges associated with establishing an effective feedback loop for developers, thereby impeding the adoption process.

4.4. Evolving Context

The third and final challenge that our analysis revealed was primarily connected to the group of caseworkers situated in Aid Center 4 that also was part of developing a new aid solution that required two different types of digital applications due to a change in government policies. However, as the content of this policy was difficult to understand, both for the development team, the caseworkers, and also governmental advisors, the development of new technology implementing these policies had become very complicated.

This led us to recognize the impact and importance of the evolving context in which the adoption process unfolds. For expert users such as caseworkers in the public sector, the context is largely connected to the implication of regulations and government policies, and it is crucial for caseworkers to have a clear understanding of these to be able to process applications from citizens. In the present context, new policies had been put in place not long before the development team had been ordered to begin the production process for digital case processing system. The aid center in question had volunteered to be part of the development process, but it was very clear that it had not turned out to be what they had imagined: *"We thought it would be two cases per year. But we are not built for two or three a day. And we thought those two cases [per year] were super easy to just turn down. But that wasn't the case, either. So it turned out to be something quite different from what we expected."*

The caseworker unit we interviewed had been informed that being part of developing the new digital solution would take a few hours yearly, but in reality, two of them were now spending a third of their time at work contributing to software development. This included two weekly online meetings with the developers and writing down all errors they encountered in the application software (which seemed to be showing up for every other case) in spreadsheets and physical notebooks. The applications fed into the wrong application system or that generally were not possible to process could only be completed together with the development team using virtual meeting software as they developed this part of the solution. The uncompleted cases were added

to a long queue of unsolved applications that remained unprocessed until they were prioritized at one of their online meetings, however, they usually only managed to go through one application per session with the team (i.e., two cases per week). During our visit to the caseworker unit, we observed a list of 42 unprocessed cases, the oldest received more than two months ago.

Since the new system was built to support new policies, the development team had to construct a completely new way of working for the caseworkers, which meant that the caseworkers first had to process applications in a near 'empty' system. The idea was that, following CSE principles, the development team would add new features incrementally, as both they and the caseworkers eventually would identify what was needed next. This way of working was, however, extremely frustrating for the caseworkers, who were not able to process any of the applications coming into the system as critical features were missing. As one caseworker said: *"Using [the new software] is like an empty room with nothing but a door in and out."* This frustration was also clearly noticeable when observing the unit at work.

Ultimately, the result of the halting development process was not only that the caseworkers felt a growing resentment for the technology development moving to fast, but they also experienced a growing frustration among the other major group of users: citizens applying for aids. Citizens frequently called caseworkers to ask advice on how to send a digital application, which one of the two systems to use, and inquiring why their applications were not yet processed as they had to wait for their devices. As it is the caseworkers' responsibility to make sure that citizens get the exact right aids and in due time, they are the ones blamed for the system not functioning. This added a lot of stress and anxiety for the caseworkers, and several in the unit were wondering why the digital solution had to be rushed through. One said: *"We would much rather have been given a completed digital system and training material to use it when [all the policy details] had been worked out."*

The frustration and guilty conscience were also visible amongst parts of the development team. They were left feeling responsible for the delayed process, despite the juridical hurdles being the main cause for the situation and would also have been impossible to predict for them. Some team members we interviewed expressed that they were doubting how well the continuous process was working in this scenario. One said: *"You must have basic functionality [in place] to get a case through by issuing money and letters, and make a decision. [...] So before you have that foundation, it should be complete in the sense that you have to have a solid foundation before you can launch. I think we launched too early."*

5. Discussion

5.1. Discussion of Findings

We began this study by asking: How is the Continuous Software Engineering development process experienced by the supply and user sides in the Norwegian public sector, and which challenges do CSE pose for the caseworker users' adoption of the software?

In CSE literature, users' trust in the product is seen as needed for them to continue using the software. This entails however that users have a choice, and that there are alternatives if trust is not "continuous" (Fitzgerald & Stol, 2017). In public sector software development, however, users (both government employees, and citizens) have no choice but to use what a governing authority provides, which reduces the importance of continuous trust.

Our findings suggest that this influenced the development team's approach to getting continuous feedback from their users, which points to the second main theme that emerged in our findings. The team admitted to not making user feedback a central part of the development process. Although this led to a sense of guilt, they saw the need to prioritize it away due to the wide range of responsibilities that permeated their work as public sector software developers, encompassing both technological and societal chores. One might hence speculate about whether a lack of profit-driven incentives discourages a focus on trust-building and continuous user communication from the developer side. Another central finding was that users became demotivated when they felt excluded or not informed about any essential information on the product under development. In plan-driven development, milestones and deadlines provide users with a sense of predictability (even though these deadlines are frequently overrun). Continuous software engineering implies that plans are defined at a much shorter planning horizon and subject to considerable change, leaving users in the dark about what will be developed and when. In contrast to both Continuous Use and extant technology adoption models, Continuous Adoption focuses on the transactional aspect of the software development process, seeing as it takes place at the point where users continuously receive and integrate new, incremental versions of the technology into their specific context. However, this study highlights how a lack of predictability may demotivate users both from providing necessary feedback to the team, and in their general attitude towards the software they had to use as their main work tool, preventing this transaction from working smoothly. Instead, caseworkers saw the continuous way of working as concealing transparency

and, relating this to findings of Mattos et al. (2020) we argue that the lack of transparency in the continuous process made it difficult for caseworkers to have a clear vision of the potential benefits of the software.

It is worth noting, however, that much of the criticism was tied to the speed of the digitalization efforts as well as the lack of extra resources to be part of the development, and not to the efforts of the development team, who were in the same situation as public sector employees. In all four centers, caseworkers saw investing time and mental effort into providing feedback to developers as an additional burden to their already heavy workload. Despite their integral role in the production of the new software, they were not formally recognized as a key stakeholder in this process, nor did their managers grant them any extra time to do so, or other privileges. Caseworkers who had to spend extra time delivering feedback due to their roles as 'super users' and contact persons for Team Welfare used their goodwill to get colleagues to cover their responsibilities for them. They thus bore the burden of spending time and cognitive labor on providing feedback, while developers gained valuable insights from users without incurring much effort themselves. This attitude seemed to stem from developers perceiving software customization as a mainly positive aspect for users, perhaps failing to see the work involved in continuously providing feedback. However, they were also not thoroughly informed about this burden from the users.

Developing software continuously for a small user base differs from developing for a large market with tens of thousands to millions, where developers may be willing to risk demotivating or even losing some users in exchange for discovering something valuable. This discovery could potentially attract many more users than they lose. However, in cases like NAV, where bespoke software is developed for a limited number of users, typically tens or hundreds, demotivating users poses a significant risk as it can greatly hinder successful software adoption and lead to negative effects such as reduced job satisfaction and possibly an increase in staff turnover. An enormous responsibility is hence put on the shoulders of the development team in that regard.

It seemed clear that both the development team and expert users would have benefited from a better mutual understanding of each other's roles and where they fit together in the complex organization to maximize their potential. These findings correspond to Klotins et al. (2022)'s argument that development teams adopting CSE can be at odds with regulatory practices in other environments, meaning that *"a careful balance between speed and discipline"* is needed. Our findings also point to how regulatory practices deriving from governmental policies directly impact and complicate software adoption. Caseworkers in Aid Center 4, who almost without their awareness had agreed to become part of developing a new digital case processing system, felt that not only had they become the development team's 'guinea pigs' in testing out 'empty' software solutions that only helped developers in determining what needed to be built next. Caseworkers in this unit also saw themselves as being forced to simultaneously execute new policies they did not understand, at the expense of citizens who relied on their provided services. This example is an instance of what Gualdi and Cordella (2022) have labeled "techno-legal entanglements"; not only does technology impact the application of public policies, policies also directly impact and complicate the development and adoption of technology.

The contextual aspect of Continuous Adoption, entailing that not just the actual use situation but also the evolving context was in constant change as a result of the incremental software delivery, also allows for discussing the evolvement of individual as well as collective adoption tactics, although that was not a focus for this study. We encountered big differences regarding not only geography and number of users for each aid center, but also in terms of team and workday organization and attitudes towards digitalization. The gradual software implementation also likely allows for different cultures in the different aid centers to be preserved. With a plan-based delivery approach, these cultural differences would likely not have been sustained in the same way. When software is delivered incrementally it is integrated into already established practices and cultures and allows for the different aid centers to develop in different, and perhaps conflicting ways. This, however, likely creates more work for the development team, and Team Welfare reported that they often received very different kinds of feedback and requests from the various aid centers around the country, as it also allows for the complexity of this user group to sustain, and even adds further complexity.

5.2. Study Limitations and Future Work

As this study addresses a specific context within the Norwegian public sector, it cannot be generalized to contexts in other countries or sectors. However, we contend that the Continuous Adoption concept is more generally relevant and extends CSE literature by highlighting the need to study the adoption of software in a continuous context, encouraging discussions around development teams' practices and implications for users' experiences. Future studies can delve further into the role of software adoption in other contexts where

development teams build software following the CSE paradigm. Of particular interest would be to study the contrast between our study of the public sector with Continuous Adoption in profit-driven teams in the private sector and their users. More research is particularly needed looking into how the entanglements of new technology and new public policies (see Gualdi and Cordella (2022)) complicate the software adoption process and create frustrations, both for the development team and their users.

Acknowledgments This research is supported by the Norwegian Research Council (grant number: 321477).

References

Auer, F., & Felderer, M. (2018). Current State of Research on Continuous Experimentation: A Systematic Mapping Study. *44th Euromicro Conference on Software Engineering and Advanced Applications*, 335–344.

Berker, T., Hartmann, M., & Punie, Y. (2005). *Domestication of media and technology*. McGraw-Hill Education.

Bosch, J., Olsson, H. H., & Crnkovic, I. (2018). It takes three to tango: Requirement, outcome/data, and AI driven development. *International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms*.

Cordella, A., & Gualdi, F. (2019). Law, technology and policies: A complex negotiation to generate value. In *3rd international conference on e-commerce, e-business and e-government* (pp. 21–28). ACM.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, *13*(3), 319–340.

Debois, P. (2011). Devops: A Software Revolution in the Making? — Cutter Consortium.

Eggebø, H. (2020). Kollektiv kvalitativ analyse. *Norsk Sosiologisk Tidsskrift*, *4*(2), 106–122.

Fabijan, A., Dmitriev, P., Olsson, H. H., & Bosch, J. (2017). The evolution of continuous experimentation in software product development. *IEEE/ACM 39th Intern. Conf. Software Engineering*, 770–780.

Fitzgerald, B. (2009). Open Source Software Adoption: Anatomy of Success and Failure. *International Journal of Open Source Software and Processes*, *1*, 1–23.

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176–189.

Gefen, D., Karahanna, E., & Straub, D. W. (2003). Trust and TAM in Online Shopping: An Integrated Model. *MIS Quarterly*, *27*(1), 51–90.

Gualdi, F., & Cordella, A. (2022). *Techno-legal entanglements as new actors in the policy-making process*.

Hoehle, H., & Huff, S. (2012). Advancing Task-Technology Fit Theory: A formative measurement approach to determining task-channel fit for electronic banking channels. In *Information Systems Foundations: Theory Building in Information Systems*. ANU Press.

Hussain, W., Perera, H., Whittle, J., Nurwidyantoro, A., Hoda, R., Shams, R. A., & Oliver, G. (2022). Human Values in Software Engineering: Contrasting Case Studies of Practice. *IEEE Transactions on Software Engineering*, *48*(5), 1818–1833.

Kaliannan, M., Awang, H., & Raman, M. (2007). Technology adoption in the public sector: An exploratory study of e-government in Malaysia. *1st International Conference on Theory and Practice of Electronic Governance*, 221–224.

Klotins, E., Gorschek, T., Sundelin, K., & Falk, E. (2022). Towards cost-benefit evaluation for continuous software engineering activities. *Empirical Software Engineering*, *27*(6), 157.

Maalej, W., Happel, H.-J., & Rashid, A. (2009). When users become collaborators: Towards continuous and context-aware user input. *24th ACM SIGPLAN OOPSLA*, 981–990.

Mattos, D. I., Dakkak, A., Bosch, J., & Olsson, H. H. (2020). Experimentation for Business-to-Business Mission-Critical Systems: A Case Study. *International Conference on Software and System Processes*, 95–104.

Meredith, J. (1998). Building operations management theory through case and field research. *Journal of Operations Management*, *16*(4), 441–454.

Moe, N. B., & Mikalsen, M. (2020). Large-Scale Agile Transformation: A Case Study of Transforming Business, Development and Operations. In *Proceedings XP 2020* (pp. 115–131). Springer.

Plesner, U., Justesen, L., & Glerup, C. (2018). The transformation of work in digitized public sector organizations. *Journal of Organizational Change Management*, *31*(5), 1176–1190.

Reddick, C. G. (2009). The adoption of centralized customer service systems: A survey of local governments. *Government Information Quarterly*, *26*(1), 219–226.

Ros, R., Bjarnason, E., & Runeson, P. (2022). A theory of factors affecting continuous experimentation (face) [arXiv:2210.05192].

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164.

Saldaña, J. (2013). *The coding manual for qualitative researchers* (2nd ed). SAGE.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557–572.

Sørensen, K. (2005). Domestication: The enactment of technology. In T. Berker, M. Hartmann, Y. Punie, & K. Ward (Eds.), *Domestication of Media and Technology*. Open University Press.

Takahashi, K., Muraoka, R., & Otsuka, K. (2020). Technology adoption, impact, and extension in developing countries' agriculture: A review of the recent literature. *Agricultural Economics*, *51*(1), 31–45.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly*, *27*(3), 425–478.

Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2019). Implementing the planning process within devops teams to achieve continuous innovation. *52nd Hawaii International Conference on System Sciences*.

Womack, J., & Jones, D. (2003). *Lean thinking: Banish waste and create wealth in your corporation* (2nd Edition). Productivity Press.

Yaman, S., Fagerholm, F., Munezero, M., Männistö, T., & Mikkonen, T. (2020). Patterns of user involvement in experiment-driven software development. *Information and Software Technology*, *120*, 106244.