Information Systems Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Navigating Temporal Plurality in Agile Software Development: A Process Explanation

Gregory Vial, Suzanne Rivard

To cite this article:
Gregory Vial, Suzanne Rivard (2026) Navigating Temporal Plurality in Agile Software Development: A Process Explanation. Information Systems Research

Published online in Articles in Advance 04 May 2026

. https://doi.org/10.1287/isre.2020.0604

https://pubsonline.informs.org/journal/isre

INFORMATION SYSTEMS RESEARCH
Articles in Advance, pp. 1–22
ISSN 1047-7047 (print), ISSN 1526-5536 (online)

Navigating Temporal Plurality in Agile Software Development: A Process Explanation

Gregory Vial,[a,*] **Suzanne Rivard**[a]

[a] Department of Information Technologies, HEC Montreal, Montreal, Quebec H3T 2A7, Canada
*Corresponding author
Contact: gregory.vial@hec.ca, https://orcid.org/0000-0001-6196-7483 (GV); suzanne.rivard@hec.ca (SR)

Received: November 21, 2020
Revised: April 4, 2022; May 16, 2023; January 18, 2024; April 29, 2025; December 3, 2025; January 23, 2026
Accepted: March 7, 2026
Published Online in Articles in Advance: May 4, 2026

https://doi.org/10.1287/isre.2020.0604

Abstract. Navigating temporal plurality—the superposition of temporal structures whose differing rhythms and phases may be desynchronized, generating temporal misfits—has long challenged software project teams, especially in agile software development (ASD), owing to its high velocity and iterativity. To shed light on this phenomenon, we conducted a longitudinal, theory-building study of five ASD projects. Anchored in the temporal structuring perspective, we mobilized the lens of entrainment to explain how actors respond to misfits created by conflicting temporal structures—activity cycles—by entraining (synchronizing with a dominant cycle or zeitgeber), resisting entrainment, or creating new temporal structures. We found that in ASD, temporal misfits emerge from four activity cycles: agile-delivery, organizational, agile-quality, and agile-communication, with the agile-delivery cycle participating in all the misfits we observed. We propose a two-stage descriptive process model of a temporal misfit episode, in which disturbance caused by the misfit first escalates through a disturbance-of-work mechanism until a team responds by (1) relinquishing the agile-delivery cycle, (2) preserving it, or (3) creating simultemporality, which enables two conflicting cycles to function in parallel. Responding reactivates the mechanism, leading to either renewed escalation with detrimental project effects or de-escalation with beneficial ones. At the project level, we suggest that the overall effect of temporal misfits and team responses emerges from a discontinuous compilation of heterogeneous individual episode impacts. Our study advances ASD research by positioning temporal plurality and its constitutive misfits at the heart of Agile teams' work and foregrounding team agency. We also advance the conversation on time in organizations by elaborating the lens of entrainment.

History: Suprateek Sarker, Senior Editor; Joe Nandhakumar, Associate Editor.

Funding: Financial support from the Social Sciences and Humanities Research Council of Canada [Grant 430-2022-00022]; the HEC Montréal Chair in Strategic Management of Information Technology; and Fonds de Recherche du Québec–Société et Culture [Grant 206971] is gratefully acknowledged.
Supplemental Material: The online appendix is available at 10.1287/isre.2020.0604.

Keywords: information systems development • entrainment perspective • time • case studies • process model • agile software development • temporal plurality

1. Introduction

Organizations implement digital technologies to enhance operations, achieve strategic goals, and maintain competitiveness. Such endeavors involve software projects—temporary initiatives well suited to reaching objectives that would otherwise be difficult, if not impossible, amid ongoing organizational activities. Nonetheless, software projects remain embedded within multiple structures that influence their unfolding. Some are external to organizations, such as field norms for IT implementation (Berente and Yoo 2012) and software development methods (Vial and Rivard 2016), while others are internal, including routines, rules, and norms (Mintzberg 1979).

A particular type of structure, temporal structures—"the organizing elements and norms that define the temporal properties of organizational systems" (Shipp and Richardson 2021, p. 301)—enable and constrain work and are intrinsic components of both internal and external structures. Navigating multiple temporal structures has long challenged software teams (e.g., Brooks 1975, Kula et al. 2022), as they operate within

temporal plurality—the superposition of temporal structures, whose differing rhythms and phases may be desynchronized, generating temporal misfits that are constitutive features of this condition. For example, in waterfall software development, late-arriving user requirements conflict with the fixed timelines of waterfall phases, negatively impacting projects (Chari and Agrawal 2018). Although such misfits are the hallmark of all software projects, they are especially critical in agile software development (ASD) due to its high-velocity, iterative time-boxed delivery using short sprint cycles (Beck et al. 2001, Vidgen and Wang 2009, Iivari 2021), while remaining embedded within broader temporal structures (O'Connor et al. 2024). Given ASD's prominence (Digital.ai 2023), it is essential to uncover the overlapping temporal structures that embed agile teams and generate misfits, how teams address those misfits, and the consequences that ensue—a question undertheorized in the information systems literature.

Our study engages with this challenge by developing a process explanation of how ASD teams navigate temporal plurality. To do so, we draw on organizational research on temporality (Bluedorn 2002, Ballard 2009, Shipp and Richardson 2021, Reinecke and Lawrence 2023, Blagoev et al. 2024) and anchor our study in the temporal structuring perspective (Orlikowski and Yates 2002), which views actors as pacing their work by drawing on temporal structures that they or others have previously enacted. Building on this, we mobilize the lens of entrainment (Bluedorn 2002, Ancona and Waller 2007, Pérez-Nordtvedt et al. 2008, Ballard 2009), aligned with temporal structuring and well-suited for investigating temporal plurality in ASD. Specifically, entrainment has served to explore contexts where actors are embedded within activity cycles—"temporal containers of work processes" (Ballard 2009, p. 24)—whose temporal requirements sometimes conflict (Orlikowski and Yates 2002, Reinecke and Ansari 2015, Blagoev and Schreyögg 2019), creating temporal misfits (Pérez-Nordtvedt et al. 2008). It also allows in-depth analysis of recurrent actors' actions in such contexts (Orlikowski and Yates 2002): entraining to a dominant cycle (*zeitgeber*) (Ancona and Waller 2007), resisting entrainment (Blagoev and Schreyögg 2019), or creating temporal structures (Shipp and Richardson 2021). Accordingly, we ask: "*How does temporal plurality manifest in ASD, how do temporal misfits arise, how do teams respond to them, and what are the project impacts of these responses?*"

Owing to the nature of our questions, we drew from a multiple-case theory-building approach to develop a theoretical explanation (Eisenhardt 1989, 2021) based on a longitudinal study of five ASD projects. Our analysis suggests four activity cycles shaping temporal plurality: agile-delivery, organization, agile-quality, and agile-communication, with the agile-delivery cycle being present in all temporal misfits we observed. Recurring misfits disturb work at each agile iteration, leading to disturbance escalation unless resolved. From our data analysis, we inductively derived three modes of responding—relinquishing the agile-delivery cycle, preserving it, and creating simultemporality—which respectively sustained, mitigated the effect, or prevented the recurrence of misfits. Assembling these elements, our primary contribution is a descriptive process model of an episode of temporal misfit, in which a disturbance-of-work mechanism operates as a feedback loop (Forrester 1968), creating disturbance escalation or de-escalation, and shaping the project impacts of a temporal misfit. Finally, we theorize the agile-delivery cycle as the zeitgeber for our ASD teams and posit that the overall project effect of temporal plurality emerges from the interplay of multiple co-occurring misfit episodes, rather than the linear accumulation of their individual impacts.

Anchored in the temporal structuring perspective (Orlikowski and Yates 2002), our study advances ASD research by positioning temporal misfits as the focal phenomenon through which temporal plurality shapes Agile teams' work, unpacking its dynamics through a process explanation. Doing so, we also contribute to research on time in organizations through the elaboration (Fisher and Aguinis 2017) of the lens of entrainment, demonstrating its applicability under ASD's high velocity and iterativity temporal conditions. We advance the contemporary conversation on entrainment and actors' agency by showing how agile teams enact agency when responding to temporal misfits and by proposing that creating simultemporality—a new temporal structure—broadens the repertoire of responses to temporal misfits.

2. Theoretical Background

Time is pervasive in software development, and the literature draws from three main perspectives to investigate it: clock time, event time, and temporal structuring. We assess the relevance of each to our study, focusing on temporal structuring, which bridges the other two perspectives and assumes temporal multiplicity in organizations (Orlikowski and Yates 2002), before introducing entrainment as an analytical lens aligned with this perspective. We argue that by foregrounding the existence of misfits among temporal structures and offering concepts for analyzing how actors respond to them, entrainment is well-suited to studying the dynamics of temporal plurality in ASD.

2.1. Time in Software Development Research

Clock time, the dominant perspective in software development research, treats time as a uniform resource with an immutable trajectory experienced by all actors

(Orlikowski and Yates 2002, p. 685). Research operationalizes it in various ways, such as time to accomplish a task (Mangalaraj et al. 2014) or time-to-market (Carmel et al. 2010). Although it provides metrics for studying efficiency and productivity, this *monotemporal* view (Nandhakumar and Jones 2001, Ballard and Seibold 2004) overlooks actors' agency in creating, interpreting, and responding to time-related indicators and fails to account for the temporal plurality that characterizes ASD. Shifting from this conceptualization, subjective time (or event time) views time as defined by actors (Orlikowski and Yates 2002), such as event-based timelines (Yakura 2002), on-time project completion (Cao et al. 2009, Lee and Xia 2010), and time overruns (Yetton et al. 2000, Pang and Lee 2022). By foregrounding actors' roles in creating and interpreting temporal objects, this perspective acknowledges that such objects may carry multiple interpretations. The notion of *pluritemporalism* further captures the assumption that actors belonging to different occupational communities assign different meanings to time or events (Hassard 1996; Yakura 2002, p. 957; Ballard and Seibold 2004). Notwithstanding its richness, this perspective provides limited insight into how actors navigate temporal plurality—understood here not as divergent interpretations of temporal objects, but as the coexistence of multiple temporal structures and their associated misfits.

The temporal structuring perspective bridges the gap between clock time and subjective time (Orlikowski and Yates 2002). From this perspective, actors routinely draw on temporal structures previously enacted by themselves or others—for example, a project schedule—to organize and pace their work. Because these structures emerge from ongoing practices, actors modify them over time. Orlikowski and Yates (2002, p. 686) refer to "the ongoing constitution of multiple temporal structures in people's everyday practices" as pluritemporalism,[1] a notion that also implies that actors face "contradictory expectations about how to temporally structure their activities" (ibid.). Importantly, although temporal structures are ongoingly constituted, temporal structuring assumes periods of stability, during which they may be treated as "objective" (Orlikowski and Yates 2002, p. 687). This assumption makes temporal structuring particularly relevant to our study, as we contend that the superposition of multiple "stabilized-for-now" (ibid.) temporal structures creates the context of temporal plurality of ASD projects. Although software development research has not extensively used temporal structuring, a few studies have nonetheless mobilized it in settings close to ASD—for instance, to investigate the effect of temporal structures embedding software developers (Stacey and Nandhakumar 2009, O'Connor et al. 2024), how virtual software development teams meet temporal challenges (Sarker and Sahay 2004), and how interactions with knowledge tools enable designers to navigate through time (Zhang et al. 2021).

Although temporal structuring acknowledges potential conflicts among temporal structures and emphasizes actors' agency in their constitution and modification, it does not provide immediate guidance in answering our research questions. Addressing them requires an analytical lens that explicitly acknowledges misfits between temporal structures, provides concepts for analyzing actors' responses, and accounts expressly for the tempo and phases of conflicting temporal structures. Within the temporal structuring perspective, the lens of temporal institutional work (Granqvist and Gustafsson 2016) possesses most of these features. Still, it focuses on long-term processes of institutional change, which are inconsistent with fast-paced temporary initiatives such as ASD projects. Another lens, temporal sensemaking (Jansen and Shipp 2019), studies the process by which actors mobilize interpretations of "past and future events" (Hernes and Obstfeld 2022, p. 2) to inform action. Yet, it does not account for temporal plurality. We identified entrainment, in its contemporary form (e.g., Reinecke and Ansari 2015), as a suitable lens: it provides concepts to characterize concurrent temporal structures, explicitly acknowledges the occurrence of temporal misfits among them, and focuses on how actors synchronize, resist, or reconfigure temporal structures, making it uniquely suited to capture the dynamics of temporal plurality in ASD.

2.2. Entrainment as an Analytical Lens

Originating in biology, entrainment explains how the environment influences the rhythms of living systems (Pérez-Nordtvedt et al. 2008) through pacers (Bluedorn 2002)—zeitgebers or time givers—that provide the temporal cues for enacting specific activities. In its original form, entrainment focuses on "the process in which the rhythms displayed by two or more phenomena become synchronized, with one of the rhythms often being more powerful or dominant and capturing the rhythm of the other" (Bluedorn 2002, p. 148). Entrainment introduces the concept of activity cycles—that is, temporal structures that are characterized by their tempo (Pérez-Nordtvedt et al. 2008, p. 785) or pace (e.g., Ancona and Waller 2007) and their phases—that is, "specific stage[s] in the cycle" (Pérez-Nordtvedt et al. 2008, p. 785) at which an activity takes—or should take—place, regardless of tempo. Because of sometimes conflicting temporal requirements among activity cycles, a corollary of synchronization is asynchrony (Mohammed and Nadkarni 2011, Blagoev and Schreyögg 2019), or temporal misfit (Pérez-Nordtvedt et al. 2008). This lens has served to theorize the role of temporal structures at different levels, including individuals (e.g., Shipp and Richardson 2021), organizations (e.g., Reinecke and Ansari 2015, Blagoev and Schreyögg 2019),

and, aligned with the focus of our work, projects (e.g., Biesenthal et al. 2015, Dille et al. 2023) and the teams conducting them (e.g., Ancona and Waller 2007).

In its original form, entrainment assumes the synchronization of actors' behavior to dominant cycles in the presence of a temporal misfit (e.g., Bluedorn 2002, Reinecke and Ansari 2015, Granqvist and Gustafsson 2016). For instance, Ancona and Waller (2007, p. 117) show teams that operate within multiple activity cycles engaging in a "dance of entrainment," alternately entraining to different cycles—"the fiscal rhythm of top management, the four-month cycle of their organizational unit, and the various cycles of their customers"—to satisfy their temporal demands. Entrainment to a dominant cycle is theorized to lead to better performance than a lack thereof (Pérez-Nordtvedt et al. 2008).

However, recent conceptual (e.g., Shipp and Richardson 2021, Reinecke and Lawrence 2023, Blagoev et al. 2024) and empirical (e.g., Reinecke and Ansari 2015, Blagoev and Schreyögg 2019) works have relaxed the assumption of automatic entrainment. Instead, they adopt the alternative assumption of actors' agency, "the capacity of actors to reflect on and influence temporal structures as they enact them" (Pentland et al. 2025, p. 119), suggesting that even entrainment may require considerable agency (Dille et al. 2023, Kunzl and Messner 2023). In brief, agentic actors adapt to, resist, or reinterpret temporal norms and rules (Pentland et al. 2025). They may contribute to stabilizing institutions as they reenact existing temporal structures (Reinecke and Lawrence 2023, p. 640), "choose" resisting entrainment (e.g., through uncoupling, as proposed by Blagoev and Schreyögg 2019), or create new structures (Shipp and Richardson 2021, pp. 308, 309, 312). They may also devise means such as ambitemporality—a process of "confrontation, reflexivity, and adaptive innovation" (Reinecke and Ansari 2015, p. 639)—to address conflicting "temporal orientations" (ibid., p. 618) of different cultural norms.

Taken together, these features make entrainment an apt lens for investigating the dynamics of temporal plurality in ASD. It builds on the concept of the activity cycle, mirroring ASD's cyclical, iterative rhythm (sprints) (Beck et al. 2001), foregrounds temporal misfits among activity cycles (Ancona and Waller 2007) central to our conceptualization of temporal plurality, and adopts an agentic assumption (e.g., Shipp and Richardson 2021), aligned with ASD's advocacy for team autonomy (Moe et al. 2021).

3. Methods

Following the principle of transparency (Sarker et al. 2013), we report the chronology of our study and key decisions. Initially, we did not focus on temporal plurality in ASD, but entered the field to gain a deep understanding of ASD practices. Early observations revealed difficulties that teams experienced at each agile iteration. These stemmed from conflicting requirements—between agile precepts (e.g., user availability) and a project's organizational context (e.g., policies about user availability), or among agile precepts (e.g., quality versus rapid software delivery). The literature provided limited explanations for the origin of such conflicting requirements, teams' responses, and their project impact. This motivated our theory-building study, which draws on Eisenhardt's multiple case study theory-building method (Eisenhardt 1989, 2021) and its core precepts (Eisenhardt 2021, pp. 149–152): address questions with limited prior evidence, engage in theoretical sampling, develop and define concepts during the analysis, advance "theoretical arguments (i.e., mechanisms)" (p. 151) to explain relationships among constructs, identify boundary conditions, address alternate explanations, and mobilize constant comparison, a logic of replication, and cross-case analysis.

3.1. Case Selection and Data Collection

We made initial contacts with two North American firms: Logistics, a small software firm, and AgileFirm, a consulting firm that uses Scrum to develop business intelligence systems. Two of the seven AgileFirm clients we approached agreed to participate: Entertain, a large entertainment firm, and Insurance, an insurance company. We selected projects through theoretical sampling to maximize variation and allow comparison. Accordingly, we selected five projects that shared a common process (Eisenhardt 2021, p. 150)—here, ASD—in "purposefully different settings" (ibid.), such as industries, firms, departments, and system types (Table 1). We studied four ongoing projects over their duration—three in Entertain and one in Logistics—and one completed project in Insurance. Table 1 presents our five cases; Online Appendix 1 provides background information.

Data collection lasted nine months and drew from multiple sources (Table 1). We conducted numerous observation sessions, taking ample notes, and carried out interviews throughout the ongoing projects up to two weeks after completion. We supplemented most interviews with email or in-person questions. We adjusted our interview protocol and guide to probe emerging themes and exploit opportunities (Eisenhardt 1989; see Online Appendix 2 for the final version of our interview guide). We recorded and transcribed the interviews with our informants' consent. Two informants denied permission, so we took detailed notes.

3.2. Data Analysis

Consistent with our theory-building approach, our data analysis was highly iterative (Eisenhardt 1989,

Table 1. Presentation of Cases and Data Sources

Case	MarketBI	EventPlan	ResourceMgmt	ScheduleMgmt	LegalBI
Organization	Entertain	Entertain	Entertain	Logistics	Insurance
Type of IS	Marketing intelligence	Operational application	Operational application	Operational application	Business intelligence
Purpose	Assist decision making	Schedule events	Manage resources	Schedule resources	Report for regulatory compliance
Client	Internal	Internal	Internal	External	Internal
Target user	Marketing analysts	Planners	Planners Resource managers	Operational staff in distribution centers	Regulatory compliance staff
Technology	Web application	Web application (cloud-based)	Web application (cloud-based)	Web application (cloud-based)	Back-office application
Official ISD method	Scrum	Scrum	Scrum	None (inspired by Scrum rituals)	None (inspired by Scrum rituals)
Sprint duration	3 weeks	2 weeks	2 weeks	2 weeks	3 weeks
Main agile rituals	Daily scrum Backlog grooming Sprint planning Sprint review Sprint retrospective	Daily scrum Backlog grooming Sprint planning Sprint review Sprint retrospective	Daily scrum Backlog grooming Sprint planning Sprint review Sprint retrospective	Sprint planning Sprint review Sprint retrospective (informal)	Daily scrum Sprint planning Sprint review Sprint retrospective (informal)
Agile process management tool	Physical (walls) Atlassian Jira	Microsoft TFS	Physical (walls) Microsoft TFS	Atlassian Jira	Physical (whiteboard)
Core team size	8	7	9	8	6
Data sources	Interviews (8) Observation (7 hours)	Interviews (5) Observation (7 hours)	Interviews (3) Observation (11 hours)	Interviews (8) Observation (5 hours)	Interviews (5, retrospective)
Informants	Client[a] Director of practice Developers (2) Functional analyst Project manager Solution architect User representative	Business analyst Developers (2) Project manager User representative[a]	Developers (2) Project manager Subject matter experts[a] (exclusively during observations)	Business analyst Developers (4) President Software architect Technical manager VP product development	Client[a] Delivery manager Quality assurance developer Lead project manager Business intelligence consultant
Data collection period[b]	03/03/2014–06/27/2014	04/17/2014–07/09/2014	11/17/2014–09/14/2015	05/26/2014–04/14/2015	12/18/2014–03/12/2015

Note. Total number of pages of interview transcripts: 665; total number of pages of field notes: 188.

[a]These respondents are future system users who were directly involved in the project, although this was always partially due to their roles in the organization.

[b]Although data collection took place between 2014 and 2015, our recent discussions with practitioners, combined with contemporary practitioner works on new approaches to ISD (e.g., Kim et al. 2021), allow us to remain confident in the currency of the phenomenon, as well as the relevance of our findings.

p. 532; Eisenhardt 2021, p. 151), as we repeatedly alternated between reading the case materials, coding, applying constant comparison, and reviewing the literature. Below, we outline the three overarching phases of this extensive and recursive analytic process.[2]

3.2.1. Phase 1: Uncovering the Role of Time. We conducted within-case analysis as we collected data. We imported our interview transcripts and observation notes into MaxQDA and performed open coding (Glaser 1978). We assessed the relevance of a given excerpt empirically—through the salience of conflicting requirements that a team faced—rather than by predetermined theoretical constructs. We identified episodes of conflicting requirements and treated each as a unit of analysis (Langley 1999). In line with constant comparison (Eisenhardt 2021, p. 152), we iterated between the data, open codes, and the patterns of relationships emerging within each case. We then conducted selective coding (Glaser 1978), comparing open codes to generate higher-level categories for conflicting requirements, team responses, and project impact. As relationships among higher-level categories emerged, we engaged in theoretical coding (Glaser 1978) and generated core concepts (Berends and Deken 2021, p. 138). We documented our analysis and conveyed our evidence of concepts and emerging relationships (Eisenhardt and Ott 2017) into data display matrices. During cross-case analysis, we compared evidence and codes across all five cases to identify differences between similar cases and similarities among different cases (Eisenhardt 1989, p. 541). This revealed time—specifically, when and at what pace activities were expected—as a common feature of the conflicting requirements our teams encountered.

3.2.2. Phase 2: Accounting for Time. Building on this serendipitous observation, we turned to the literature on time, identifying temporal structuring as our theoretical perspective and entrainment as our analytical lens. We re-engaged with our data and purposefully searched for empirical evidence of activity cycles, generating corresponding codes. We re-entered within-case and cross-case analyses, systematically revisiting the raw data, our data display matrices, open codes, higher-level categories, and core concepts, applying the vocabulary of entrainment where appropriate. This led us to recast our phenomenon of interest from "conflicting requirements among entities" to "temporal misfits among temporal structures," namely, activity cycles. We observed that the impact of a temporal misfit and the associated response was not immediate but followed a recurrent pattern: a rise in work disturbance at each iteration until the team responded, followed by either an increase or a decrease in disturbance in subsequent iterations. To capture this, we revisited the data, updated our data display matrices (Online Appendix 3), and conducted open and selective coding anew. We then undertook another iteration of theoretical coding from which our final core concepts (Tables 2 and 3 and Online Appendix 4) and preliminary theoretical explanation emerged.

3.2.3. Phase 3: Theorizing Time. Consistent with earlier iterations that treated an episode of temporal misfit as a unit of analysis, we began our theorizing by zooming in on an episode. We developed a process model that articulated the origin, unfolding, and impact of a temporal misfit in ASD. Although our fundamental understanding of the phenomenon we studied remained stable, the model evolved over multiple data analysis iterations involving constant comparison, seeking to identify the mechanism underlying the process (Eisenhardt 2021), leading us to refine our concept definitions and explanations. We then moved to a higher level of abstraction, asking why temporal misfits occur in ASD. This prompted us to revisit the literature on temporal structuring and entrainment, leading us to propose the concept of *temporal plurality*, a condition in which potential misfits exist among "stabilized-for-now" (Orlikowski and Yates 2002, p. 687) temporal structures. We refined our research questions accordingly: *How does temporal plurality manifest in ASD, how do temporal misfits arise, how do teams respond to them, and what are the project impacts of these responses?* We then zoomed out from the episode level to theorize temporal plurality in ASD at the project level.

4. Findings

We introduce our core concepts and bring them to life with project narratives. Table 2 documents the empirical grounding of these concepts, linking illustrative evidence, open codes, and high-level categories to core concepts. Online Appendix 3 presents our data-display matrices for each uniquely identified episode of temporal misfit (denoted with using hashtags), and Online Appendix 4 reports the complete evidence. The emerging concepts serve as theoretical signposts (Berends and Deken 2021, p.140) that our model will later connect. We present them in the sequence of our research questions, mirroring the unfolding of an episode of temporal misfit derived from our analysis.

4.1. Agile-Delivery Temporal Misfit

We begin by describing how temporal misfits emerged across activity cycles. Entrainment views organizational entities as embedded in multiple activity cycles that set temporal work requirements, among which misfits may arise (e.g., Bluedorn 2002, Ancona and Waller 2007, Blagoev and Schreyögg 2019). We identified four activity cycles immediately relevant to our

Table 2. Empirical Grounding of Core Concepts

Illustrative quote excerpts	Open code	*Higher-level category*	*CORE CONCEPT*
"The emphasis was set on having iterative development[…] what mattered most was to be able to show something to the client as early as possible." (Developer 2, EventPlan)	Early delivery	*Speed of delivery*	***AGILE-DELIVERY CYCLE***
"If after two weeks you haven't managed to deliver what you were supposed to deliver, you ask yourself why [you could not]." (Project manager, MarketBI)	Fast delivery		
"[Our sprints lasted] three weeks. Actually, we used to do three weeks, then we moved to two weeks." (Developer 3, EventPlan)	Sprint duration	*Sprint characteristics*	
"It's a lot easier to follow [when we use agile sprints], I mean, everyone has a capacity, everyone is assigned tasks." (Developer 2, ResourceMgmt)	Capacity planning		
"My commitment, it's the first priority that I need to deliver during that month. […] We need to deliver this, we will deliver it. After that, we get into other priorities." (VP product development, ScheduleMgmt)	Prioritization		
"The project had to be delivered at a fixed date and with fixed functionality [as dictated by the PMO]. Therefore, on Insurance's side, there was a lot less agility." (Client, LegalBI)	PMO phases & pace	*Organization imposed schedule parameters*	***ORGANIZATION CYCLE***
"I think it is inevitable. […] It's the [external] client who pays [based on an expected delivery date] and when the client has money, at the end of the day, a check, it drives things quite a bit." (Architect, ScheduleMgmt)	Schedule impediments due to management		
"Some team members were more, not distant, but not available 100% of the time [due to commitments from the organization]." (Developer 1, ResourceMgmt)	Limited resource availability	*Resource constraints*	
"Sometimes it takes a month before you have the answer from the client." (Project manager, MarketBI)	Client delay		
"We wanted to deliver value so much that we would rather spend a week building a dashboard, building a report, cleaning data than to write documentation." (Director of practice, MarketBI)	Generating value for customers	*High quality solution attributes*	***AGILE-QUALITY CYCLE***
"In business intelligence, you can […] deliver a report without any sort of [sound] architecture, anyway […] when you do things really well, you adopt something like a data vault approach, with all required steps, really, your product is super clean." (Delivery manager, LegalBI)	Implementing sound architecture		
"We [business analysts and members of the implementation team] also have [testing] blitzes. VP product development can, at some point say: 'Alright we have to move on to another stage, we will do a testing blitz.' We each have a little query we run on Jira and we find everything that needs to be tested." (Business analyst, ScheduleMgmt)	Focusing on task completion	*Supporting processes for high quality solution*	
"Every time we would have arguments and discussions [between developers] and be stubborn about different things. But the result is always well done, it's always well crafted." (Developer 2, EventPlan)	Collective problem-solving		
"Everything related to discussing lessons during a sprint, about agility, that happens during the sprint review. Other than that, it's very rare." (Project manager, MarketBI)	Sprint rituals as communication platforms	*Communication rituals*	***AGILE-COMMUNICATION CYCLE***

Table 2. (Continued)

Illustrative quote excerpts	Open code	*Higher-level category*	*CORE CONCEPT*
Observation notes and informal discussions with ResourceMgmt team members indicate that on the last day of every sprint (every other Friday), the team met during the entire morning to perform a sprint review, followed by sprint planning, and a demo of the current version to project stakeholders. Halfway through every sprint (every other Monday), the team met for two hours to prepare the next sprint (backlog grooming).	Time required for sprint rituals		
"The first [communication] mechanism is face to face communication. Agility facilitates cubicle-less environments […] so people talk to one another. Literally, if there is a problem right now, the first question is 'Do you have five minutes? No. Ok, when? In twenty minutes. Perfect, I will get back to you then.'" (Project manager, MarketBI)	Availability for communication	*Enabling communication*	
"When we have a question, we turn around and we talk to each other. […] Proximity to people [also helped a lot for communication." (Developer 2, EventPlan)	Collocation for real-time communication		
"The job of product owner [that the user was supposed to assume] is hands-on, it's about someone being available. The resource has to be dedicated to the project … but really full-time. *[…] [The level of participation of the user] was very low. We see it, for example in the acceptance tests, they did the bare minimum.*" (Business analyst, EventPlan)/(#1 - EP_1)	Speed vs. resource availability	*Agile-delivery cycle/organization cycle misfit*	***AGILE-DELIVERY TEMPORAL MISFIT***
"We considered ourselves to be alone at the battlefront. They [Entertain's PMO] assigned DB Architect to do the database […]. *In the end, those [including the DB Architect] that were supposed to be sitting in the room with us were not in the room, so I ended alone with the other developer.*" (Developer 1, ResourceMgmt)/(#2 - RS_2)	Self-organization vs. centralized control		
"*In the final analysis, the budget was set based on the scope*, except that the scope changed in the course of the project." (Architect, MarketBI)/(#8 - MB_3)	Responding to change vs. efficiency and effectiveness		
"*As early as January, we had [implementation] people setting up a new client who wanted to go live with the new version* […] six months before we had even completed our first release [according to the product roadmap]." (VP product development, ScheduleMgmt)/(#7 - SC_2)	Organization calendar vs. team calendar		
"At one point your [SME] role is really to give us a guideline on [the requirements], *and the business analyst is supposed to arrive and say, 'Yeah, I understand your guideline, except that the business is going this way [referring to requirements demanded by the organization's project management office], so we'll compromise.*'" (Developer 2, ResourceMgmt)/(#18 - RS_4)	Demands of organization vs. demands of team		
"*We can't just sort of say, 'OK, we'll drop that [database] table, we'll replace it because we've found a new way to do it.*' It's too late for that." (Architect, ScheduleMgmt)/(#3 - SC_3)	Speed vs. quality	*Agile-delivery/Agile-quality misfit*	
"*All the same, there's a cost to just … all the planning, all those sorts of things …* that really consumes meeting time, even though you wouldn't think it would." (QA lead, LegalBI)/(#4 - LG_3)	Speed vs. communication	*Agile-delivery/Agile-communication misfit*	

Table 2. (Continued)

Illustrative quote excerpts	Open code	*Higher-level category*	*CORE CONCEPT*
"You tend to want to answer the client right away and soon it gets completely out of hand. We found ourselves in [daily] scrums lasting a half hour, forty minutes." (QA lead, LegalBI)/(#4 - LG_3)	Excessive internal communication	*Recurring delays*	***DISTURBANCE ESCALATION***
"I have to report on my global scope every week and when my user tells me: 'No, no, forget about it, in the end I am not interested in this [data] source anymore, we focus elsewhere,' I come to my steering [committee] and I say: 'No we changed the global scope.' Wow, that raises a big red flag, the sirens, the whole shebang." (Project manager, MarketBI)/(#8 - MB_3)	Excessive reporting with outsiders		
"So the testing was delayed, delayed." (Frontend developer, MarketBI)/(#5 - MB_1)	Waiting for users/clients		
"Every time we would ask for a build from the technical team, it would take a while." (Senior business analyst, ScheduleMgmt)/(#6 - SC_1)	Waiting for experts		
"We had to use CVS to put our code in production. […] But the first part of the project, we did not have CVS, we only knew that it was coming eventually." (QA Lead, LegalBI)/(#10 - LG_2)	Waiting for tools		
"In the case of [Entertain], it took about two months before [the peer review process] got implemented." (Project manager, MarketBI)/(#9 - MB_4)	Waiting for process		
"Every month, we would pretty much say: 'Another two months and the new version will be released, I can't believe it.' 'I can't believe it,' that was our typical line: 'I can't believe that it won't be over.' But it wasn't over. So, overall, there weren't many positive points. Negative points, there are many of them. […] Every time we had to backport new code into older versions, always fixing bugs, with our clients who want to go live with this version that isn't even completed." (VP product development, ScheduleMgmt)/(#7 - SC_2)	Waiting for developers		
"Some users […] would come back to tell me that the system was not answering all their needs. Had they participated every week, done all the required testing and all that kind of stuff, they could have realized that, and we could have made it right from the beginning." (User representative, EventPlan)/(#1 - EP_1)	System quality issues for users	*Recurring quality issues*	
"We do not have a QA team, we have an implementation team. And right now, the implementation team spends 90% of their time configuring new customer environments, so there is 10% of their time dedicated to testing. And these tests, at the end of the day, the bugs are discovered while configuring these environments." (Architect, ScheduleMgmt)/(#3 - SC_3)	Code (architectural) quality issues		
"The only way to contain it [waiting for architects' decisions] was to have [Project manager] do tight follow-ups, stating: 'Look, we have a deadline, the integration must work by this date but we depend on this [integration] project and we cannot finish until we have it.' And eventually, it would escalate." (Developer 2, ResourceMgmt)/(#17 - RS_3)	Escalating issues to higher authorities	*Entraining (to organization cycle)*	***RELINQUISHING***
"We always knew that something [referring to the work of DB Architect] would modify our way of coding. It can be something small, it can be something big, but that's what we don't know yet." (Developer 1, ResourceMgmt)/(#2 - RS_2)	Waiting for input from expert		

Table 2. (Continued)

Illustrative quote excerpts	Open code	*Higher-level category*	*CORE CONCEPT*
"We had a little backlog of questions. We would tell [the external proxy], 'Here are our questions' […] and she would run tests and then send it to the user who would say 'Oh no, this is not what we want.'" (Delivery manager, LegalBI)/(#15 - LG_1)	Having a nonteam member involved on an occasional basis		
"We [Logistics] had made commitments that we had to honor and we had no other option, so we released code. Code that was not ready to go live." (VP product development, ScheduleMgmt)/(#7 - SC_2)	Honoring project commitments		
Agile management allows us to make decisions on our own, without consulting [the steering committee], without them really noticing, for their own good." (Project manager, MarketBI)/(#8 - MB_3)	Hiding information to side with the user	*Entraining (to agile-delivery cycle)*	***PRESERVING***
"You hide that from the client big time. Big time! At one point he'll say, 'Why don't you have 40 days, why do you have only 30 days for this sprint?' 'There is something that should be redone, trust me, it'll get redone.' You hope that the conversation ends there." (Project manager, MarketBI)/(#9 - MB_4)	Hiding information to enable learning		
We had our ETL [extract, transform and load] objects all the same, they were on a server, at a single location with a proper name. And so what we set up was that if someone needed to work on a … on the ETL object that was in the directory, he would use his prefix—we had given ourselves abbreviations for our usernames … . He used a prefix for the directory." (QA lead, LegalBI)/(#10 - LG_2)	Using software to escape reliance on others	*Temporal bridging*	***CREATING SIMULTEMPORALITY***
"In terms of TFS [a tool to manage software project activities and source code], that's really where we were monitoring our tasks." (Developer 1, ResourceMgmt)/(#22 - RS_5)	Using software to replace face-to-face communication		
"We set up a board, with three columns." (Backend developer, LegalBI)/(#4 - LG_3)	Using physical objects to replace face-to-face communication		
"Yes. And at first it was the sprint 0, we created standardized dimensions. […] So, understand the basics of the solution's architecture. […] A solution blueprint, we call it." (Director of practice, MarketBI)/(#20 - MB_5)	Developing a blueprint to guide development		
"Very much [business analyst] who served as an interface with [users]." (Developer 2, EventPlan)/(#1 - EP_1)	Having a team member play the role of user		
"When we finally received the message: 'Ok, here's how it's going to work' from the architecture team, I spent one sprint, one sprint and a half tearing down our database model to reconstruct it [according to DB Architect's guidelines]." (Developer 1, ResourceMgmt)/(#2 - RS_2)	Accumulation of delays	*Recurring negative moderated impact on project*	***POSTRESPONSE DISTURBANCE ESCALATION***
"The architects would pass the message along to [another team member], and when he went to their meetings, often he had a different opinion from theirs. […] They did not agree with him. At some point they stopped inviting him to those meetings. [Raises his voice, expressing frustration] Seriously, they stopped inviting him and they never allowed developers to attend those meetings." (Developer 2, ResourceMgmt)/(#17 - RS_3)	Accumulation of rework		

Table 2. (Continued)

Illustrative quote excerpts	Open code	*Higher-level category*	*CORE CONCEPT*
"I told [Project manager]: 'Look, it's your perception and the developers are telling you the project is done, but you should have come to see the client to see what we were doing with that money before you went back to the investment committee.' […] His credibility really took a hit because he went to the investment committee to get a smaller amount, only to go back at the end of the project [to ask for additional funds]." (Business analyst, EventPlan)/(#14 - EP_3)	Fewer features delivered at the end of each iteration	*Recurring negative moderated impact on system*	
"It was the worst summer of my life. Yes, really. I left for vacation. […] And starting from the week I came back, which was the first week of July, I worked 80 hour weeks until the end of August. I was burnt out." (VP product development, ScheduleMgmt)/(#7 - SC_2)	Accumulation of stress	*Recurring negative moderated impact on actors*	
"When I have fewer questions to answer [from the project management office], I have time to focus on my team, to make sure they deliver." (Project manager, MarketBI)/(#12 - MB_2)	Less external influence on team's work and ability to learn	*Recurring positive moderated impact on learning*	***POSTRESPONSE DISTURBANCE DE-ESCALATION***
"[Business analyst] would go see the users, […] and then would make decisions […]. When he performed tests, we got feedback right away and we knew what we had to change." (Developer 1, EventPlan)/(#1 - EP_1)	Reduction in wait time and delays	*Recurring positive moderated impact on time management*	
"[During his turn at the daily scrum] Business analyst tells the team that he has found fewer issues and that he has been testing the application very thoroughly." (Observation notes, EventPlan)/(#1 - EP_1)			
"We use JIRA to manage bugs and everything. […] Once the client started using it, we saw a difference because they entered bug reports on their own, there was a list of issues to follow up on. So once [JIRA] was in place, it was great." (Director of practice, MarketBI)/(#21 - MB_6)	Less time required to communicate project progress and plan		
Every observation of agile rituals that involved multiple actors throughout the ResourceMgmt project (e.g., sprint planning, grooming, and sometimes daily scrums) relied on [Project manager] showing TFS on a large screen to analyze, discuss, and adjust the team's workload. (Observation notes, ResourceMgmt)/(#22 - RS_5)	Less time spent interacting and explaining during meetings		
"It's being able to always respond faster, to have a new version faster, to be able to get ahead faster." (Senior business analyst, ScheduleMgmt)/(#6 - SC_1)	Less time spent on noncore activities		
"We were a bit isolated from the client [referring to the organization] who has their own methodologies […], a number of processes that we did not have to comply with. We managed to isolate ourselves from all that." (Developer, LegalBI/(#10 - LG_2)	Less wait time		
"It cost a lot. […] And it's expensive; we're talking about $200,000 or more, easily. […] ResourceMgmt has spent virtually a quarter of its budget on the architecture, maybe even more." (Developer 2, ResourceMgmt)/(#17 - RS_3)	Project costs overruns	*Negative impact for project*	***DETRIMENTAL IMPACT***
"There was a part that was extremely important. […] That report ended up being prioritized at the end of the project, and we had to rush to complete it." (Developer 2, ResourceMgmt)/(#16 - RS_1)	Project deadlines slippages		

Table 2. (Continued)

Illustrative quote excerpts	Open code	*Higher-level category*	*CORE CONCEPT*
"[The architecture team] made our lives very difficult. We were spinning our wheels." (Developer 2, ResourceMgmt)/(#2 - RS_2)	Reduced morale	*Negative impact for actors*	
"I seriously considered leaving [the company]." (VP product development, ScheduleMgmt)/(#7 - SC_2)			
"There are so many things that we did not have time to test ourselves, so … in terms of quality we are behind." (Backend developer, MarketBI)/(#5 - MB_1)	Lower system quality	*Negative impact for system*	
"So basically, it allowed us have some confidence that if it was green, I hadn't broken the code for someone else." (Developer 1, EventPlan)/(#19 - EP_4)	Improved morale	*Positive impact for actors*	***BENEFICIAL IMPACT***
"At the end of the project, people are going to come back and ask about your initial commitment. Did we go where we wanted, did we deliver what we were supposed to deliver? […] My client [referring to the user] is happy with the other end of the solution, so I've got an ally." (Project manager, MarketBI)/(#12 - MB_2)	Improved user satisfaction		
"The validation was 95% done by the time we got in production; we were very confident." (Backend developer, MarketBI)/(#9 - MB_4)	Higher system quality	*Positive impact for system*	
"When you are in a context like ours, where we do our own deployments and so on, it [a continuous integration system] is huge." (President, ScheduleMgmt)/(#6 - SC_1)	Compliance with project deadlines	*Positive impact for project*	

Notes. Evidence for agile-delivery temporal misfits is highlighted, for each quote: the text associated with the agile-delivery cycle in bold and the text associated with the other activity cycle in italics. Evidence for disturbance escalation, postresponse disturbance escalation, and de-escalation is provided with underlined text that reflects the temporal, recurring aspect of impacts.

Table 3. Core Concepts Definitions

Core concept	Definition	Nature of the concept
Agile-delivery cycle	Temporal structure driven by the objective of delivering value to customers in the form of working software at frequent, regular intervals.	Temporal structure
Organization cycle	Temporal structure driven by organization-wide policies that control the schedule and timing for the enactment of specific activities.	
Agile-quality cycle	Temporal structure characterized by the slow rhythm demanded for building, maintaining, and testing products to deliver high-quality, working software.	
Agile-communication cycle	Temporal structure characterized by specific phases within an iteration during which team members are expected to perform a variety of communication rituals.	
Agile-delivery temporal misfit	Lack of synchronization between the agile-delivery cycle and another activity cycle.	Lack of synchronization between temporal structures
Disturbance escalation	Recurring detrimental effect of an agile-delivery temporal misfit that disrupts teamwork and accumulates at each iteration until the team responds to the misfit.	Recurring impact of misfit
Relinquishing	Responding by entraining to the organization cycle, at the expense of the agile-delivery cycle.	Responding to misfit
Preserving	Responding by actively seeking to remain entrained to the agile-delivery cycle, at the expense of the other cycle.	
Creating simultemporality	Responding by mobilizing people or artifacts—IT or other—to generate a new temporal structure that preserves the pace and phases of both cycles involved in an agile-delivery temporal misfit, enabling their concurrent functioning.	
Postresponse disturbance escalation	Recurring effect of an ASD team responding to an agile-delivery temporal misfit, occurring at each agile iteration, adding to the misfit's detrimental effect.	Recurring moderated impact of responding
Postresponse disturbance de-escalation	Recurring effect of an ASD team responding to an agile-delivery temporal misfit, occurring at each agile iteration, subtracting from the misfit's detrimental effect.	
Detrimental impact	Ultimate negative effect of postresponse disturbance escalation associated with responding to an agile-delivery temporal misfit on the project's outcome.	Final impact of misfit on project
Beneficial impact	Ultimate positive effect of postresponse disturbance de-escalation associated with responding to an agile-delivery temporal misfit on the project's outcome.	

ASD teams (Table 2 and Online Appendix 4, Tables A4.1–A4.4): agile-delivery, organization, agile-quality, and agile-communication. We observed 22 episodes of temporal misfit between cycles, all of which implicated the fast-paced agile-delivery cycle (Online Appendix 4, Table A4.5), structured around the delivery of software after a time-boxed iteration (sprint), and influencing how work is phased and paced: "If after two weeks you haven't managed to deliver what you were supposed to deliver, you ask yourself why" (Project manager, MarketBI). This rhythm often conflicted with the organization cycle, timed around organizational policies on resource availability, reporting schedules, or delivery dates: "Our [client representative] complained that they had to work on other projects [as dictated by Insurance]" (Project manager, LegalBI). It also differed from the demands of the agile-quality cycle, which depends upon the time required to deliver high-quality software in terms of both user experience and system architecture: "We would ask ourselves, 'Before we send the files to the customer … wouldn't it better for us to run some sort of tests to ensure that there is no unsatisfaction on her end?'" (Delivery manager, LegalBI). Similarly, its temporal demands diverged from those of the agile-communication cycle, cadenced by frequent rituals (e.g., team daily meetings and user demos): "The scrum meetings we had every morning, that was pretty good. […] We knew what everyone was working on and that helped a lot for communication" (Developer 2, EventPlan). In light of this, *agile-delivery temporal misfit* emerged as our focal concept.

All five teams experienced an agile-delivery temporal misfit with the organization cycle. For instance, the EventPlan (#1) team expected the active participation of a product owner—a user—critical for delivering "working software frequently," as per the Agile principle. Yet, given its own cycle, Entertain made the user available only part-time, conflicting with the team's delivery tempo. A business analyst observed:

> The job of the product owner is hands-on; it's about someone being available. The resource must be dedicated to the project … really full-time. […] [The level of participation of this user] was very low, for example, […] in the acceptance tests. […] When you play multiple roles, you cannot perform tests as thoroughly, even though this is where users are most useful.

In ScheduleMgmt (#6) and ResourceMgmt (#2), technical staff unavailability due to the organization cycle caused temporal misfits. The developers depended on architects assigned to guide technical design decisions. The team expected them to be reachable as needed, as their timely decisions were critical to project execution. Yet, the architects were only sporadically available because Entertain assigned them to many projects, some more urgent than ResourceMgmt, compromising the team's ability to get timely guidance.

Temporal misfits also arose between the agile delivery and the other agile cycles. The fast-paced agile-delivery cycle conflicted with the slower agile-quality cycle, as observed in ScheduleMgmt (#3), MarketBI (#20), and EventPlan (#19). In ScheduleMgmt, frequent deliveries conflicted with the time required to ensure quality. The Architect observed: "We can't just sort of say, 'OK, we'll drop that [database] table, we'll replace it because we've found a new way to do it.' It's too late for that." In ResourceMgmt (#11, #22), MarketBI (#21), and LegalBI (#4), misfits occurred between agile-delivery and agile-communication cycles. In LegalBI, for instance, frequent client communications and Agile team rituals—although fostering participation and engagement—jeopardized development time and the delivery of a working system every few weeks.

4.2. Disturbance Escalation

Temporal misfits caused delays and quality issues, disrupting work. Their recurrence led to *disturbance escalation* (Table 2 and Online Appendix 4, Table A4.6). In LegalBI (#4), recurring delays occurred due to the temporal misfit between agile-delivery and agile-communication. The QA Lead recounted recurring delays when the team attempted to address client issues during meetings: "You tend to want to answer the client right away, and soon it gets completely out of hand. We found ourselves in [daily] scrums lasting a half hour, 40 minutes … ." Recurring delays also occurred when agile-delivery temporal misfits with the organization's cycle were due to user or technical staff unavailability. The MarketBI (#5) frontend developer deplored user unavailability, noting: "the testing was delayed, delayed" because users were seldom available for feedback at iteration ends. In ScheduleMgmt (#6), developers frequently needed technical experts' support to build, deploy, and configure the system. Yet, these experts were often unavailable due to urgent matters, repeatedly delaying version availability and timely testing. A business analyst recalled: "Every time we would ask for a build from the technical team, it would take a while."

Recurring system quality issues occurred in EventPlan (#1, #3, #19), ScheduleMgmt (#3), and MarketBI (#20). For instance, EventPlan's (#1) agile-delivery cycle was based on a two- to three-week sprint. Still, the organization cycle prevented the team from securing ongoing user participation. As a result, system quality could not be evaluated by users regularly and was often lacking. The primary user representative, who repeatedly tried to enlist the other users' active participation for the duration of the project, recalled:

> Some users did not take the time to get involved as much as necessary […] They would return to tell me the system was not meeting their needs. Had they participated every week, done all the required testing, and all that kind of stuff, they could have realized that, and we could have made it right from the beginning.

ScheduleMgmt (#3) experienced recurring quality issues due to a misfit between the agile-delivery cycle and the quality cycle. Indeed, the team made architectural choices primarily driven by the need to deliver software, devoting almost 90% of its resources to new code development, leaving little for system testing. This resulted in errors discovered late in the process "while configuring [the user] environments" (Architect, ScheduleMgmt).

4.3. Relinquishing, Preserving, Creating Simultemporality: Responding to Misfit

As their work was disrupted by the recurring impacts of agile-delivery temporal misfits, our teams took three different types of actions to address them (Table 2 and Online Appendix 4, Tables A4.7, A4.8, and A4.9).

4.3.1. Relinquishing. Instances of *relinquishing*—that is, entraining to the organization cycle at the expense of the agile-delivery cycle—occurred in all five cases. For example, the ResourceMgmt (#2) team had been delayed over several iterations, waiting for guidelines from a database architect whose participation in the project was limited due to other commitments per the organization cycle's requirements. Although the team

tried to work as if not depending on the architect's availability, they continued waiting for their input:

> We always knew that something [referring to the work of DB Architect] would modify our way of coding. It can be something small, it can be something big, but that's what we don't know yet. (Developer 1).

ScheduleMgmt (#7) also experienced an agile-delivery temporal misfit with the organization cycle, as they had to meet the delivery deadline set by management based on agreements with external clients. This conflicted with building the system incrementally and iteratively, aligned with the team's roadmap. The team entrained to the organization cycle, rushing a release: "We [Logistics] had made commitments we had to honor, and we had no other option, so we released code. Code that was not ready to go live." (VP product development)

4.3.2. Preserving. Teams sometimes responded by *preserving*—actively seeking to remain entrained to the agile-delivery cycle. The MarketBI (#8, #9, and #12) team did so repeatedly when faced with a misfit with the organization cycle. In one episode (#8), Entertain's project management office (PMO) tried to enforce regular meetings for the project manager to inform the organization's steering committee of the project status. This conflicted with the agile-delivery cycle, under which the team could reprioritize work at every iteration to satisfy customers. The project manager hid information from the steering committee to remain focused on users' needs. In another (#9), he did the same to protect his team's reflection time to learn and improve their work processes:

> You hide that from the client [PMO] big time. Big time! At one point he'll say, 'Why don't you have 40 days, why do you have only 30 days for this sprint?' 'There is something that should be redone, trust me, it'll get redone.' You hope that the conversation ends there.

4.3.3. Creating Simultemporality. In various instances, all five teams responded by *creating simultemporality*—that is, mobilizing people or artifacts (IT or other) to generate a new temporal structure that preserved the pace and phases of both cycles involved in a misfit, enabling their concurrent functioning. LegalBI (#10) experienced an agile-delivery temporal misfit when the need to release data models and data ingestion pipelines frequently conflicted with Insurance's organization cycle, whose phases delayed the team's access to software critical to timely version control and deployment. The team created simultemporality by implementing their own versioning and deployment procedures.

The ResourceMgmt (#11) team experienced an agile-delivery temporal misfit with the communication cycle, being repeatedly disturbed by outsiders who visited and asked impromptu questions about the project, delaying work. Responding to this misfit, the team organized their "war room" walls into sections where slides, images, and sticky notes documented each iteration, informing outsiders (Online Appendix 3 and Online Appendix 4, Table A4.9).

EventPlan (#1) initially experienced quality issues due to a temporal misfit between agile delivery and the organizational cycle, as the former required ongoing user participation, whereas the latter maintained users' regular work commitments. The team created simultemporality by appointing their business analyst as a proxy user who gathered extensive information from users to the point where they became an apt substitute. One developer referred to the business analyst as follows: "[Business analyst] who served as an interface with [users]."

4.4. Postresponse Disturbance Escalation/ De-escalation

Because of ASD's high iterativity, the effect of teams' responding to an agile-delivery temporal misfit recurred at every agile iteration. This postresponse, recurring moderated impact manifested as disturbance escalation or de-escalation, adding to or subtracting from the misfit's detrimental effect (Table 2 and Online Appendix 4, Tables A4.10 and A4.11).

4.4.1. Postresponse Disturbance Escalation. Recall that in ResourceMgmt (#2), work depended on the guidance of a database architect unavailable as frequently as required by the agile-delivery cycle. The team relinquished the agile-delivery cycle, waiting for the architect, leaving them in a position where whatever work they had done would need rework once the architect validated it. This led to *postresponse disturbance escalation* through the compounding of rework, which could only be addressed toward the project's expected delivery date. One developer recalled spending a significant amount of time redoing work from several iterations earlier:

> When we finally received the message: 'OK, here's how it's going to work' from the architecture team, I spent one sprint, one sprint and a half tearing down our database model to reconstruct it [along DB Architect's guidelines] […] That is work for which the ROI is rather limited. […] And it cost to the user in some cases.

In ScheduleMgmt (#7), the team relinquished the agile-delivery cycle and entrained to Logistics' organization cycle by releasing code prematurely to honor the delivery date set by management. They then engaged in frantic bug fixing to render the system usable. With each iteration, the team experienced increasing stress as they felt their target and schedule

were constantly disrupted by the need to fix a system that should never have been released. Logistics' VP of product development recalled that this period was highly stressful for everyone:

> It was the worst summer of my life […] I was burnt out. […] It was a morbid summer spent trying to make our active projects survive, to continue development, to try and close a version that we could not close, and to continue supporting sales.

4.4.2. Postresponse Disturbance De-escalation. Responding could curb disturbance escalation. For MarketBI's project manager, preserving by hiding information from top management (#8, #9, and #12) shielded the team from external demands and helped keep focus on delivering within the iterations established at the project's onset: "When I have fewer questions to answer [from the PMO] I have time to focus on my team, to make sure they deliver" (#12, MarketBI Project manager).

The EventPlan team created simultemporality to preserve the pace of the agile-delivery cycle and Entertain's organization cycle by having their business analyst act as a user substitute, consistently reducing wait time for developers (#1). This positive effect was in stark contrast to LegalBI, where a similar agile-delivery temporal misfit was addressed by enlisting a user representative not part of the project team, thus entraining to the organization cycle (see Online Appendix 3, #15). Because this person had ongoing commitments outside the project, the result was "a lot of latency" (LegalBI's delivery manager) rather than de-escalation.

ScheduleMgmt (#6) had been repeatedly disturbed by the unavailability of technical staff with responsibilities outside the project. The team created simultemporality between the agile-delivery and the organization cycles by using a continuous integration system (CIS) to automatically build, deploy, and test the system whenever developers published code. This allowed bypassing the wait for technical staff's support and "respond faster, to have a new version faster, to be able to get ahead faster" (Logistics' Senior business analyst).

4.5. Detrimental/Beneficial Impact on Project

All three modes of responding (relinquishing, preserving, and creating simultemporality) impacted the project outcome (Table 2 and Online Appendix 4, Tables A4.12 and A4.13). The ultimate impact of each response depended on whether postresponse disturbance escalation or de-escalation followed.

4.5.1. Detrimental Impact. The ResourceMgmt team experienced agile-delivery temporal misfits with Entertain's organization cycle due to the unavailability of technical experts (see Online Appendix 3, #2 and #17). Developers responded by relinquishing the agile-delivery cycle, waiting for the experts' availability. This led to postresponse disturbance escalation as delays accumulated during subsequent iterations. Ultimately, three detrimental impacts were observed, as recalled by a developer: (1) Team members experienced much frustration that impacted their morale, even after completing the project. They felt that they "were spinning [their] wheels" because their velocity had constantly suffered from this situation; (2) it led to cost overruns: "It cost a lot. […] And it's expensive; we're talking about $200,000 or more, easily"; and (3) it prevented delivery as per the original roadmap, as the project was one month late. Similarly, in ScheduleMgmt (#7), the detrimental impacts of relinquishing the agile-delivery cycle were increased stress levels, leading some to question continuing to work at Logistics even after project completion. The VP product development vividly remembered that he and other team members "seriously considered leaving [the company]."

4.5.2. Beneficial Impact. In contrast, in the same project, the final impact of the ScheduleMgmt team (#6) responding to a misfit between the agile-delivery cycle and the organization cycle by creating simultemporality using a CIS was positive, so much so that Logistics kept the CIS in place even after project completion. The Logistics President commented: "[I]n a context like ours, where we do our own deployments … it [a CIS] is huge."

Recall that MarketBI's Project manager preserved the agile-delivery cycle, protecting his team from the temporal requirements of Entertain's organization cycle (#12), hiding information from the steering committee, leading to postresponse disturbance de-escalation. The manager knew that this would likely raise issues after project completion. Still, he was willing to face those because of the positive impact on user satisfaction:

> At the end of the project, people will come back and ask about your initial commitment. Did we go where we wanted, did we deliver what we were supposed to deliver? You will have to answer those questions. But you know what? I fight this fight at the end of the project. You [the organization] want to hit me? Hit me as much as you want. My client [referring to the user] is happy with the other end of the solution, so I've got an ally.

Together, the core concepts presented above describe the recurring patterns we observed across episodes of temporal misfit. Table 3 summarizes their definitions before we integrate them into a process explanation.

5. Navigating Temporal Plurality in ASD: A Theoretical Explanation

Addressing our first research question, we found that in the projects we studied, temporal plurality

manifests through four activity cycles—*agile-delivery, organization, agile-quality, and agile-communication*—that are sometimes desynchronized (Blagoev and Schreyögg 2019), imposing conflicting temporal requirements and creating *temporal misfits* (Pérez-Nordtvedt et al. 2008). Notably, the agile-delivery cycle was part of all misfits. To address our remaining questions, we explain how temporal misfits arose, how teams responded, and the project impacts that followed. We first zoom in on an episode of temporal misfit, weaving our core concepts into a descriptive process model and proposing a mechanism[3] underlying these dynamics (Cornelissen 2017, p. 6; Cloutier and Langley 2020, p. 2). As shown in our data (see Section 4, Section 4.4, and Online Appendices 3 and 4, Tables A4.6 and A4.7), each agile iteration involving a temporal misfit or a team response was followed by either disturbance escalation or de-escalation. We theorize that these recurring patterns are due to a disturbance-of-work mechanism operating through a feedback loop (Forrester 1968), in which each iteration reinforces or mitigates prior disturbance. We then zoom out to temporal plurality across the whole project.

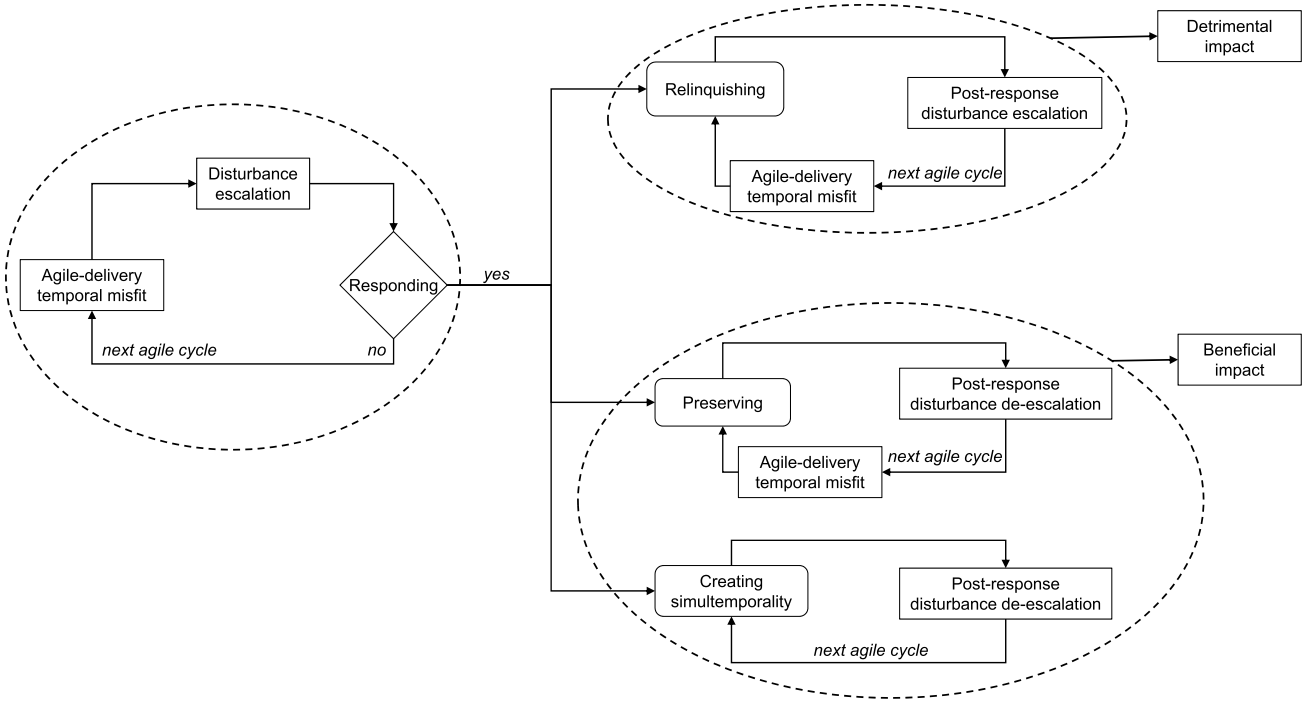
5.1. Zooming In: The Unfolding of an Episode of Agile-Delivery Temporal Misfit

An episode of temporal misfit spans the time between the first occurrence of a specific misfit and its ultimate impact on the project. Figure 1 illustrates our model, developed through constant comparison across all 22 episodes. We abstracted our teams' means of responding—relinquishing, preserving, or creating simultemporality—and integrated them and their observed effects into a descriptive process model, thereby connecting episode-level evidence to concept formation, then to our theoretical explanation grounded in the entrainment lens.

5.1.1. Initial Stage: Disturbance Escalation. A *temporal misfit* is associated with suboptimal performance, as it disturbs teamwork, causing delays or software quality issues. Left unaddressed, the misfit recurs at each agile iteration, triggering a positive feedback loop (Forrester 1968) in the disturbance-of-work mechanism, leading to *disturbance escalation*. This is illustrated by LegalBI experiencing recurring delays due to an agile-delivery temporal misfit with the agile communication cycle (#4) or ScheduleMgmt experiencing recurring quality issues (#3) owing to an agile-delivery temporal misfit with the quality cycle—see Section 4. As shown in Online Appendices 3 and 5, disturbance escalation followed temporal misfits in all 22 episodes we observed.

5.1.2. Critical Event: Responding. Over time, the level of work disturbance reaches a point where the team responds to the misfit to restore progress. For our teams, responding took one of three forms. The first is *relinquishing*, with the team entraining to the organization cycle, at the expense of the agile-delivery cycle. The second is *preserving*, in which the team prioritizes entrainment to the agile-delivery cycle, at the expense of the other cycle. The third is *creating simultemporality*, in which the team mobilizes people or artifacts—such

Figure 1. Descriptive Process of the Unfolding of an Agile-Delivery Temporal Misfit Episode



as IT or other resources—to generate a temporal structure aimed to preserve the pace and phases of both cycles involved in the misfit and enable their concurrent functioning. Responding—a manifestation of the team's agency to address a temporal misfit (e.g., Shipp and Richardson 2021, Reinecke and Lawrence 2023)—marks a critical inflection point in the trajectory of disturbance escalation, ending the initial stage and initiating one of two alternative second stages.

5.1.3. Alternative Second Stage: Postresponse Disturbance De-escalation. We theorize that both creating simultemporality and preserving trigger a negative feedback loop within the disturbance-of-work mechanism, followed by *postresponse disturbance de-escalation* and *beneficial impacts*, though differently. In our teams, preserving helped in maintaining the tempo and phases of the agile-delivery cycle and neutralizing the misfit without preventing its recurrence. In contrast, by mobilizing people or artifacts, simultemporality prevented temporal misfit recurrence, allowing both cycles to maintain their rhythm. Whereas in one episode (#8), the MarketBI project manager withheld information from Entertain's steering committee to maintain customer focus, preserving the agile-delivery's tempo and phases despite recurring misfit, EventPlan (#1) responded differently. The team appointed a business analyst as a proxy for unavailable users, creating simultemporality that allowed maintaining the pace of both the agile-delivery and organization cycles, averting further misfits, enabling concurrent activities, and ultimately being followed by disturbance de-escalation.

5.1.4. Alternative Second Stage: Postresponse Disturbance Escalation. We propose that relinquishing triggers the positive feedback loop within the disturbance-of-work mechanism anew, adding to work disturbance as the team entrains away from the agile-delivery cycle at each agile iteration, creating *postresponse disturbance escalation*. Over time, *detrimental impacts* on the project, including reduced morale, accumulation of technical debt, and delays, follow. For instance, in ResourceMgmt (#2), waiting for a database architect repeatedly led to rework, escalating disturbance, and threatening the project's delivery schedule. Similarly, in ScheduleMgmt (#7), releasing code prematurely to meet organizational deadlines resulted in frantic bug fixing, increased stress, and reduced morale. These, and other episodes (Online Appendices 3 and 5), show that relinquishing the agile-delivery cycle is associated with detrimental project impacts.

5.2. Zooming Out: How Temporal Plurality Affects the ASD Project

Having described the unfolding of an episode of agile-delivery temporal misfit, we now extend our theorizing to the project level. First, anchored in the entrainment lens, we posit the agile-delivery cycle as the *zeitgeber* (Bluedorn 2002, Ancona and Waller 2007, Ballard 2009) (time-giver) for the ASD teams we studied, as it was present in all the temporal misfits we observed. Across our cases, maintaining the team entrained to this cycle was associated with postresponse disturbance de-escalation and beneficial project impacts. In contrast, relinquishing was followed by disturbance escalation and detrimental impacts. These findings are consistent with prior theorizing, suggesting that compliance with the temporal requirements of the zeitgeber is associated with higher performance (Pérez-Nordtvedt et al. 2008) and underscore the dominance of the agile-delivery cycle—rooted in the Agile Manifesto—in our empirical context.

Second, espousing the assumption of actors' agency (Blagoev and Schreyögg 2019, Shipp and Richardson 2021, Reinecke and Lawrence 2023), we unpack its role in team responding. In small structures embedded within larger organizational contexts, entraining to an activity cycle originating from an external institution (in our study, the Agile Manifesto) at the expense of the organizational cycle entails agency and demands effort (Dille et al. 2023, Kunzl and Messner 2023). Consider the case of MarketBI, where the project manager withheld information from the organization. Although aware that this action carried long-term personal and organizational risks, he assumed responsibility and chose preserving the agile-delivery cycle. Creating simultemporality—a new temporal structure (Orlikowski and Yates 2002, Shipp and Richardson 2021)—also required agency, as teams devised interventions that reconciled conflicting cycles by mobilizing people or artifacts, such as the implementation of a CIS by the ScheduleMgmt team, enabling the organization and the agile-delivery cycles to operate at their own rhythm.

Relinquishing—resisting (Blagoev and Schreyögg 2019, Pentland et al. 2025)—the agile-delivery cycle and, for our teams, entraining to the organization cycle also entail agency. Although this strategy may appear sensible because it involves entraining to a legitimate, hierarchically higher activity cycle, it can lead to detrimental project impacts as it fosters the escalation of disturbance. For example, in ResourceMgmt, the project manager at times responded to temporal misfits by entraining to the organization cycle—for example, waiting for architects' availability to make critical design decisions, even though this hampered the team's progress and was followed by significant negative consequences for the team members, who suffered delays and reduced morale.

Third, we found that ASD's high velocity and iterativity, combined with temporal plurality, create a context in which work is frequently impeded by multiple episodes of temporal misfit, implying the multiplication

of the process we theorize. Furthermore, as shown in Online Appendix 5, our teams did not respond uniformly to all misfits they encountered. At times, they created simultemporality in response to a given temporal misfit and engaged in a trajectory of disturbance de-escalation with beneficial impacts, yet after responding to another misfit, engaged in a trajectory of disturbance escalation after relinquishing, yielding detrimental impacts. For instance, MarketBI enacted preserving twice and relinquishing once when facing agile-delivery temporal misfits with the organization cycle and created simultemporality twice in response to misfits with another agile cycle. Each responding mode was followed by either disturbance escalation with detrimental impacts or de-escalation with beneficial impacts. This suggests that the overall project effect of temporal misfits and team responses is a multilevel construct emerging from the aggregation of individual-level episode impacts. Our findings indicate that this aggregation is not an isomorphic linear composition of homogeneous elements, but rather, a discontinuous compilation of heterogeneous individual episode impacts (Klein and Kozlowski 2000).

6. Discussion

6.1. Contribution to ASD: Explaining How Teams Navigate Temporal Plurality

Our work expands knowledge on ASD in two ways. First, we explicitly engage with the role of time—acknowledged as central to ASD but seldom theorized (O'Connor et al. 2024). Time is indeed an essential aspect of this prevalent mode of software development (Digital.ai 2023), with the literature frequently referring to concepts such as velocity (e.g., Dikert et al. 2016, Elbanna and Sarker 2016, Iivari 2021), momentum (e.g., Vidgen and Wang 2009), and tools such as burndown charts (e.g., Hoda et al. 2011, Dennehy and Conboy 2019). We focus on the misfits that arise between the temporal requirements of the agile-delivery cycle—which align with the high velocity and iterativity institutionalized in the Agile Manifesto (Beck et al. 2001)—and those of three other activity cycles—organization, agile-quality, and agile-communication. Together, these four cycles constitute the context of temporal plurality surrounding the agile teams we studied. We highlight the pre-eminence of the agile-delivery cycle and position it as the zeitgeber within our empirical context. Although research acknowledges velocity and iterativity as critical (e.g., Conboy 2009, Dikert et al. 2016, O'Connor et al. 2024), we foreground their theoretical significance by portraying their prominence through the unfolding of an episode of agile-delivery temporal misfit, within which we identify a disturbance-of-work etiological mechanism (Ylikoski 2013) that traces the causal chain from the initial occurrence of a misfit through its recurrence across agile iterations, team responding, and ensuing project impacts. Additionally, we reveal the theoretical relevance of temporal misfits between agile-delivery and other agile activity cycles—communication and quality. Indeed, although studies have examined contradictions in ASD (e.g., Moe et al. 2010, Iivari 2021), ours is the first to theorize their unfolding and impacts through a temporal lens. Although developed within an ASD context, we surmise that our explanation extends to the broader software development phenomenon, where time-based tensions have been a critical issue (e.g., Wiedemann et al. 2023).

Second, we leverage the assumption of actors' agency to explain how our teams navigated temporal plurality by responding to agile-delivery temporal misfits. Both research (e.g., Gregory et al. 2015, Iivari 2021) and practice (e.g., Beck et al. 2001) emphasize the benefits of autonomy in ASD, which enables teams to make crucial decisions independently (e.g., Moe et al. 2019, 2021). Our theoretical explanation illuminates how these benefits materialized for our agile teams, whether through preserving the agile-delivery cycle or creating simultemporality—a new, sustainable temporal structure that reconciles conflicting temporal requirements. It also draws attention to situations in which teams relinquished the agile-delivery cycle, to the detriment of their project. At the project level, we show that teams exercise their agency through all three means of responding, with varying degrees of success, nuancing the taken-for-granted benefits of team autonomy enshrined in the Agile Manifesto.

6.2. Contributing to Research on Time in Organizations Through the Study of ASD

Building on our empirical analysis, we contribute to the elaboration of the entrainment lens (Fisher and Aguinis 2017) in two ways. First, we refine understanding of actors' agency, showing that our ASD teams exercise it by entraining to the zeitgeber (preserving), resisting it (relinquishing), or creating a new temporal structure. We foreground creating simultemporality as a novel manifestation of agency, in which actors mobilize people or artifacts to preserve the pace and phases of both cycles involved in a misfit, enabling their concurrent functioning. Metaphorically, people or artifacts function as a *differential gear* in an automobile—"a device fitted to the axle of a vehicle that allows the wheels to turn at different rates"[4]—preserving the smooth operation of the vehicle. Similarly, the people or artifacts that a team mobilizes to create simultemporality enable two conflicting activity cycles to maintain their respective rhythm without perturbing the operation of either. The metaphor of a differential gear should not be conflated with Yakura's (2002) *temporal boundary object*, whose function is intrinsically different. Temporal boundary objects—such as timelines—are "essentially narrative

representations that allow diverse groups to fill in content and interpretations and negotiate as they see fit […] They provide a locus for communication, conflict, and coordination" (Yakura 2002, p. 968). In contrast, the differential gear is performative: it is a material component of a new temporal structure that allows conflicting cycles to operate concurrently. Rather than mediating between temporal perspectives, it reconfigures two distinct temporal structures into a new one.

Creating simultemporality is conceptually distinct from several other means of responding to temporal misfits found in the literature. Because it allows both activity cycles involved in a misfit to maintain their own rhythm, it differs from the dance of entrainment (Ancona and Waller 2007), where teams alternately modify their rhythm to entrain to several activity cycles. It is also distinct from creating ambitemporality—a process of reworking temporal tensions "through confrontation, reflexivity, and adaptive innovation" (Reinecke and Ansari 2015, p. 639)—and from related concepts developed through lenses other than entrainment, such as temporal ambidexterity, which refers to "firms' attempts to balance their short-term and long-term needs" (Slawinski and Bansal 2015, p. 544). We contend that creating simultemporality is an act of temporal structuring (Orlikowski and Yates 2002), through which teams build a stabilizing temporal structure that reconfigures rather than mediates temporal conflicts. These distinctions position creating simultemporality as a unique enactment of actors' agency.

Second, we extend the applicability of the entrainment lens to temporal conditions that differ from extant research. Most previous empirical work took place in contexts where entrainment—or its absence—unfolded over extended periods (e.g., four decades in Blagoev and Schreyögg 2019 and implicitly several years in Reinecke and Ansari 2015) or over shorter periods still significantly longer than the two- to three-week iterations typical of ASD (four months in Ancona and Waller 2007). Our study shows that entrainment can be mobilized in highly iterative and veloce environments, revealing temporal misfits, their associated responses, and the ensuing impacts.

Overall, our study of how temporal plurality gives rise to misfits, how teams respond to them, and with what project consequences is aligned with calls to "seek the mechanisms of social entrainment" (Orlikowski and Yates 2002, p. 696) by studying "the recurrent actions of individuals establishing or reinforcing the temporal structures that are being 'captured' (or, for that matter, the reified structures that are 'capturing' them)" (ibid.).

6.3. Implications for Practice

Our work has implications for practitioners. First, our processual view of temporal plurality invites ASD teams to consider both the immediate and longer-term effects of temporal misfits and their responses. Indeed, although ASD typically focuses on the sequencing of activities *within* each time-boxed iteration (sprint), our work extends this view to illuminate how decisions made within those iterations impact the entire duration of the project. Second, we offer creating simultemporality as a means of reconciling conflicting activity cycles. Although responses that favor one activity cycle may be more readily implemented, creating simultemporality is a suitable alternative that meets the temporal requirements of both. Although this response may call for resources and time to actualize its potential, it can enable a sustainable path to managing conflicting temporal requirements and even to be reused across projects, as observed in ScheduleMgmt with the implementation of a CIS or, more generally, with the implementation of continuous integration and delivery practices advocated by modern agile approaches such as DevOps (e.g., Wunderlich et al. 2019, Kim et al. 2021).

6.4. Research Avenues

As a theory-building endeavor, our study has limitations that present opportunities for future research. Although we acknowledge that *multiple* episodes of agile-delivery temporal misfits occur concurrently, we did not explore potential interactions among episodes. Future research could advance our work by examining the potential compounded effects of multiple episodes. We investigated traditional agile contexts, where the prevailing approach was Scrum. Although this is consistent with most current ASD projects, future research could study temporal misfits in other contexts. Recent approaches (e.g., DevOps) advocating the seamless integration of activities across units have gained prominence, and there is evidence that achieving this high degree of integration is difficult, particularly as teams possess what Wiedemann et al. (2023, p. 10) refer to as different "time rhythms." In our study, technological artifacts were instrumental to the emergence of simultemporality. Future research could examine whether similar patterns are observed with these new approaches that mobilize technology to automate and monitor software delivery and operations.

7. Conclusion

Our study illuminates how ASD teams navigate temporal plurality—the coexistence of overlapping temporal structures that generate recurring misfits. Drawing on longitudinal evidence from five projects, we developed a process explanation that accounts for how temporal misfits arise, how teams respond to them, and the ensuing project impacts. Our findings extend the entrainment lens to highly iterative settings

and foreground actors' agency, especially in the creation of a new temporal structure—simultemporality—to reconcile conflicting cycles. In practice, our study suggests that agile teams should anticipate the combined effects of temporal misfits and consider creating simultemporality to enable the smooth operation of otherwise conflicting cycles.

Acknowledgments

The authors are grateful to the senior editor, associate editor, and anonymous reviewers for their highly insightful comments and suggestions throughout the entire review process.

Endnotes

[1] Although Yakura (2002) and Orlikowski and Yates (2002) use the same term, they propose different concepts. Orlikowski and Yates' (2002) concept of pluritemporalism is also different from that of temporal plurality, which is our empirical context.

[2] Table 2 presents selected evidence illustrating each core concept, and Table 3 lists the core concepts' definitions. Quotes detailing the unfolding of each episode of temporal misfit are available in Online Appendix 3. Complete evidence for all codes is available in Online Appendix 4.

[3] Cloutier and Langley's (2020) and Cornelissen's (2017) conceptualization of mechanism is similar to Avgerou's (2013), which we adopt here: a mechanism is a set "of entities and activities that produce change from an initial state to observed outcomes" (Avgerou 2013, p. 407). Although the concept is central to critical realism, it is not restricted to this movement (Hedström and Ylikoski 2010).

[4] See https://dictionary.cambridge.org/dictionary/english/differential-gear.

References

Ancona D, Waller MJ (2007) The dance of entrainment: Temporally navigating across multiple pacers. Rubin BA, ed. *Workplace Temporalities* (Emerald Group Publishing Limited, Bingley, UK), 115–146.

Avgerou C (2013) Social mechanisms for causal explanation in social theory based IS research. *J. Assoc. Inform. Systems* 14(8):399–419.

Ballard DI (2009) Organizational temporality over time: Activity cycles as sources of entrainment. Roe RA, Waller MJ, Clegg SR, eds. *Time in Organizational Research* (Routledge, New York), 238–254.

Ballard DI, Seibold DR (2004) Organizational members' communication and temporal experience: Scale development and validation. *Commun. Res.* 31(2):135–172.

Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Highsmith J, et al. (2001) Manifesto for Agile software development. Accessed March 1, 2023, http://agilemanifesto.org/.

Berends H, Deken F (2021) Composing qualitative process research. *Strategic Organ.* 19(1):134–146.

Berente N, Yoo Y (2012) Institutional contradictions and loose coupling: Postimplementation of NASA's enterprise information system. *Inform. Systems Res.* 23(2):376–396.

Biesenthal C, Sankaran S, Pitsis T, Clegg S (2015) Temporality in organization studies: Implications for strategic project management. *Open Econom. Management J.* 2(1):45–52.

Blagoev B, Schreyögg G (2019) Why do extreme work hours persist? Temporal uncoupling as a new way of seeing. *Acad. Management J.* 62(6):1818–1847.

Blagoev B, Hernes T, Kunisch S, Schultz M (2024) Time as a research lens: A conceptual review and research agenda. *J. Management* 50(6):2152–2196.

Bluedorn AC (2002) *The Human Organization of Time: Temporal Realities and Experience* (Stanford University Press, Stanford, CA).

Brooks FP (1975) *The Mythical Man-Month* (Addison-Wesley Longman, Boston).

Cao L, Mohan K, Xu P, Ramesh B (2009) A framework for adapting agile development methodologies. *Eur. J. Inform. Systems* 18(4): 332–343.

Carmel E, Espinosa JA, Dubinsky Y (2010) "Follow the sun": Workflow in global software development. *J. Management Inform. Systems* 27(1):17–38.

Chari K, Agrawal M (2018) Impact of incorrect and new requirements on waterfall software project outcomes. *Empirical Software Engrg.* 23(1):165–185.

Cloutier C, Langley A (2020) What makes a process theoretical contribution? *Organ. Theory* 1(1), https://doi.org/10.1177/2631787720902473.

Conboy K (2009) Agility from first principles: Reconstructing the concept of agility in information systems development. *Inform. Systems Res.* 20(3):329–354.

Cornelissen J (2017) Editor's comments: Developing propositions, a process model, or a typology? Addressing the challenges of writing theory without a boilerplate. *Acad. Management Rev.* 42(1):1–9.

Dennehy D, Conboy K (2019) Breaking the flow: A study of contradictions in information systems development (ISD). *Inform. Tech. People* 33(2):477–501.

Digital.ai (2023) 17th state of Agile report. Accessed October 5, 2025, https://info.digital.ai/rs/981-LQX-968/images/RE-SA-17th-Annual-State-Of-Agile-Report.pdf.

Dikert K, Paasivaara M, Lassenius C (2016) Challenges and success factors for large-scale agile transformations: A systematic literature review. *J. Systems Software* 119:87–108.

Dille T, Hernes T, Vaagaasar AL (2023) Stuck in temporal translation? Challenges of discrepant temporal structures in interorganizational project collaboration. *Organ. Stud.* 44(6):867–888.

Eisenhardt KM (1989) Building theories from case study research. *Acad. Management Rev.* 14(4):532–550.

Eisenhardt KM (2021) What is the Eisenhardt Method, really? *Strategic Organ.* 19(1):147–160.

Eisenhardt KM, Ott TE (2017) Rigor in theory building from multiple cases. Raza M, Jain S, eds. *Routledge Companion to Qualitative Research in Organization Studies* (Routledge, New York), 79–91.

Elbanna A, Sarker S (2016) The risks of agile software development: Learning from adopters. *IEEE Software* 33(5):72–79.

Fisher G, Aguinis H (2017) Using theory elaboration to make theoretical advancements. *Organ. Res. Methods* 20(3):438–464.

Forrester JW (1968) Industrial dynamics—After the first decade. *Management Sci.* 14(7):398–415.

Glaser BG (1978) *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory* (Sociology Press, Mill Valley, CA).

Granqvist N, Gustafsson R (2016) Temporal institutional work. *Acad. Management J.* 59(3):1009–1035.

Gregory P, Barroca L, Taylor K, Salah D, Sharp H (2015) Agile challenges in practice: A thematic analysis. *Agile Processes Software Engrg. Extreme Program. XP 2015 Internat. Conf. Agile Software Development* (Springer, Cham, Switzerland), 64–80.

Hassard J (1996) Images of time in work and organization. Clegg SR, Hardy C, Nord W, eds. *Handbook of Organization Studies* (Sage Publications, Thousand Oaks, CA), 581–598.

Hedström P, Ylikoski P (2010) Causal mechanisms in the social sciences. *Annu. Rev. Sociol.* 36:49–67.

Hernes T, Obstfeld D (2022) A temporal narrative view of sensemaking. *Organ. Theory* 3(4), https://doi.org/10.1177/26317877221131585.

Hoda R, Noble J, Marshall S (2011) The impact of inadequate customer collaboration on self-organizing Agile teams. *Inform. Software Tech.* 53(5):521–534.

Iivari J (2021) A paradox lens to systems development projects: The case of agile software development. *Commun. Assoc. Inform. Systems* 49, https://doi.org/10.17705/1CAIS.04901.

Jansen KJ, Shipp AJ (2019) Fitting as a temporal sensemaking process: Shifting trajectories and stable themes. *Human Relations* 72(7):1154–1186.

Kim G, Humble J, Debois P, Willis J (2021) *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations* (IT Revolution, Portland, OR).

Klein KJ, Kozlowski SWJ (2000) *Multilevel Theory, Research, and Methods in Organizations: Foundations, Extensions, and New Directions* (Jossey-Bass, San Francisco).

Kula E, Greuter E, Van Deursen A, Gousios G (2022) Factors affecting on-time delivery in large-scale agile software development. *IEEE Trans. Software Engrg.* 48(9):3573–3592.

Kunzl F, Messner M (2023) Temporal structuring as self-discipline: Managing time in the budgeting process. *Organ. Stud.* 44(9): 1439–1464.

Langley A (1999) Strategies for theorizing from process data. *Acad. Management Rev.* 24(4):691–710.

Lee G, Xia W (2010) Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quart.* 4(1):87–114.

Mangalaraj G, Nerur S, Mahapatra R, Price KH (2014) Distributed cognition in software design: An experimental investigation of the role of design patterns and collaboration. *MIS Quart.* 38(1):249–274.

Mintzberg H (1979) *The Structuring of Organizations* (Prentice Hall, Englewood Cliffs, NJ).

Moe NB, Dingsøyr T, Dybå T (2010) A teamwork model for understanding an agile team: A case study of a Scrum project. *Inform. Software Tech.* 52(5):480–491.

Moe NB, Šmite D, Paasivaara M, Lassenius C (2021) Finding the sweet spot for organizational control and team autonomy in large-scale agile software development. *Empirical Software Engrg.* 26(5):101.

Moe NB, Dahl BH, Stray V, Karlsen LS, Schjødt-Osmo S (2019) Team autonomy in large-scale agile. *Proc. Annu. Hawaii Internat. Conf. System Sci. HICSS* (AIS Electronic Library, Maui, HI), 6997–7006.

Mohammed S, Nadkarni S (2011) Temporal diversity and team performance: The moderating role of team temporal leadership. *Acad. Management J.* 54(3):489–508.

Nandhakumar J, Jones M (2001) Accounting for time: Managing time in project-based teamworking. *Accounting Organ. Soc.* 26(3): 193–214.

O'Connor M, Conboy K, Dennehy D, Carroll N (2024) Temporal complexity in information systems development flow: Challenges and recommendations. *Commun. Assoc. Inform. Systems* 54:699–735.

Orlikowski WJ, Yates J (2002) It's about time: Temporal structuring in organizations. *Organ. Sci.* 13(6):684–700.

Pang M-S, Lee G (2022) The impact of IT decision-making authority on IT project performance in the U.S. federal government. *MIS Quart.* 46(3):1759–1776.

Pentland BT, Kremser W, Goh KT (2025) Path nets: Concurrence and recurrence in the dynamics of organizing. *Acad. Management Rev.* 50(1):114–137.

Pérez-Nordtvedt L, Payne GT, Short JC, Kedia BL (2008) An entrainment-based model of temporal organizational fit, misfit, and performance. *Organ. Sci.* 19(5):785–801.

Reinecke J, Ansari S (2015) When times collide: Temporal brokerage at the intersection of markets and developments. *Acad. Management J.* 58(2):618–648.

Reinecke J, Lawrence TB (2023) The role of temporality in institutional stabilization: A process view. *Acad. Management Rev.* 48(4):639–658.

Sarker S, Sahay S (2004) Implications of space and time for distributed work: An interpretive study of US–Norwegian systems development teams. *Eur. J. Inform. Systems* 13(1):3–20.

Sarker S, Xiao X, Beaulieu T (2013) Qualitative studies in information systems: A critical review and some guiding principles. *MIS Quart.* 37(4):iii–xviii.

Shipp AJ, Richardson HA (2021) The impact of temporal schemata: Understanding when individuals entrain versus resist or create temporal structure. *Acad. Management Rev.* 46(2):299–319.

Slawinski N, Bansal P (2015) Short on time: Intertemporal tensions in business sustainability. *Organ. Sci.* 26(2):531–549.

Stacey P, Nandhakumar J (2009) A temporal perspective of the computer game development process. *Inform. Systems J.* 19(5):479–497.

Vial G, Rivard S (2016) A process explanation of the effects of institutional distance between parties in outsourced information systems development projects. *Eur. J. Inform. Systems* 25(5):448–464.

Vidgen R, Wang X (2009) Coevolving systems and the organization of agile software development. *Inform. Systems Res.* 20(3):355–376.

Wiedemann A, Wiesche M, Gewald H, Krcmar H (2023) Integrating development and operations teams: A control approach for DevOps. *Inform. Organ.* 33(3):100474.

Wunderlich P, Veit DJ, Sarker S (2019) Adoption of sustainable technologies: A mixed-methods study of German households. *MIS Quart.* 43(2):673–691.

Yakura EK (2002) Charting time: Timelines as temporal boundary objects. *Acad. Management J.* 45(5):956–970.

Yetton P, Martin A, Sharma R, Johnston K (2000) A model of information systems development project performance. *Inform. Systems J.* 10(4):263–289.

Ylikoski P (2013) Causal and constitutive explanation compared. *Erkenntnis* 78:277–297.

Zhang Z, Yoo Y, Lyytinen K, Lindberg A (2021) The unknowability of autonomous tools and the liminal experience of their use. *Inform. Systems Res.* 32(4):1192–1213.

Gregory Vial is an associate professor of information technology at HEC Montreal. His research interests are in the areas of software development, data governance and management, and digital phenomena. His work has been published in *Information Systems Research*, *MIT Sloan Management Review*, *Journal of Strategic Information Systems*, *Information Systems Journal*, *IEEE Software*, and others.

Suzanne Rivard is a professor of information technology at HEC Montreal. She is a Fellow of the Royal Society of Canada, as well as a LEO Award recipient and Fellow of the Association of Information Systems. She received her PhD from the Ivey School of Business, the University of Western Ontario and holds honorary doctorates from Aix-Marseille Université and Université de Lausanne. Her work has been published in such journals as *MIS Quarterly* and *Organization Science*.