Information Systems Research

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

Translating Agile Development in Different Institutional Contexts: Dynamism, Pluralism, and Ratcheting over Time

Nicholas Berente, Sean Hansen, Christoph Rosenkranz, Sanja Tumbas

To cite this article:
Nicholas Berente, Sean Hansen, Christoph Rosenkranz, Sanja Tumbas (2026) Translating Agile Development in Different Institutional Contexts: Dynamism, Pluralism, and Ratcheting over Time. Information Systems Research

Published online in Articles in Advance 11 Mar 2026

. https://doi.org/10.1287/isre.2021.0352

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

INFORMATION SYSTEMS RESEARCH
Articles in Advance, pp. 1–15
ISSN 1047-7047 (print), ISSN 1526-5536 (online)

Translating Agile Development in Different Institutional Contexts: Dynamism, Pluralism, and Ratcheting over Time

Nicholas Berente,[a] Sean Hansen,[b] Christoph Rosenkranz,[c,*] Sanja Tumbas[d]

[a] Mendoza College of Business, University of Notre Dame, Notre Dame, Indiana 46556; [b] Boler College of Business, John Carroll University, University Heights, Ohio 44118; [c] University of Cologne, 50923 Cologne, Germany; [d] Institute for Digital Technology Management, Bern University of Applied Sciences, 3005 Bern, Switzerland

*Corresponding author

Contact: nberente@nd.edu, https://orcid.org/0000-0002-1403-4696 (NB); shansen@jcu.edu, https://orcid.org/0000-0003-0519-5815 (SH); rosenkranz@wiso.uni-koeln.de, https://orcid.org/0000-0003-4286-4176 (CR); sanja.tumbas@bfh.ch, https://orcid.org/0000-0002-2072-7843 (ST)

Received: July 4, 2021
Revised: January 21, 2022; February 9, 2023; February 11, 2024; February 13, 2025; January 15, 2026
Accepted: January 24, 2026
Published Online in Articles in Advance: March 11, 2026

https://doi.org/10.1287/isre.2021.0352

Abstract. Agile development has emerged as the dominant approach to developing software across much different institutional environments. We report a study of how organizations in four different institutional contexts translate agile development in different ways. Drawing on institutional logics and translation theory, we highlight how the dynamism and pluralism of the institutional environment influence the ways that agile development is translated by organizations. For example, contexts higher in pluralism move to increased documentation, whereas contexts marked by less dynamism move toward longer cycle times. Further, we identify a process of "ratcheting" for how these organizations translate agile methods in a stepwise, accumulative manner away from canonical agile practices over time.

History: Suprateek Sarker, Senior Editor; Ning Su, Associate Editor.

Funding: This work was supported by the National Science Foundation [Grant 1240160].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/isre.2021.0352.

Keywords: agile software development • translation theory • institutional logics • qualitative research • institutional aspects of information systems • management of IS projects • grounded theory

Introduction

Agile software development (hereafter, "agile"; Beck et al. 2001) is the most widely used approach for creating software, and is particularly important in the digital age, where software is essential for many products, processes, and business models (Barroca et al. 2019, Berente 2020). Yet, agile is not a one-size-fits-all approach; different organizations implement agile in different ways (Baham and Hirschheim 2022). Organizations translate the concept of agile into practices that fit their institutional contexts (Pries-Heje and Baskerville 2017).

Existing literature emphasizes the initial steps in this translation process—the decision-making through which organizations customize agile methods, incorporating some (but typically not all) agile practices (Cao et al. 2009). This decision-making step is often described as "tailoring" (Avison and Fitzgerald 2003, Fitzgerald et al. 2006). Tailoring involves choices made to implement agile to align with organizational, project, and team-related factors (e.g., Iivari and Iivari 2011, Tolfo et al. 2011, Campanelli et al. 2018). Initial tailoring is not the end of the process—agile practices continue to evolve after adoption through a translation process that is shaped by an organization's institutional context (Pries-Heje and Baskerville 2017). Institutional pressures that organizations face, such as normative standards, regulatory constraints, and stakeholder demands (Baxter et al. 2023), over time reshape how agile is practiced across different organizations. For example, a government development team may introduce more documentation and approval requirements over time in response to regulatory oversight (Neumann et al. 2024), whereas a tech startup might drop even relatively lightweight Scrum rituals to move faster (Giardino et al. 2016). Although existing research indicates that institutional context contributes to the translation of agile over time (Pries-Heje and Baskerville 2017), there is no

theory of how different institutional contexts shape this translation.

Different institutional contexts may involve very different institutional logics (Friedland and Alford 1991). Institutional logics describe the rationalities of organizations operating in different sectors of society and can involve many different goals, assumptions, normative standards, and material practices that provide the templates and scripts for organizational action (Thornton et al. 2012). They mark different organizational contexts and inform how the translation process occurs (Nielsen et al. 2014). After all, a government agency typically follows a different institutional logic than a tech startup. However, different institutional contexts are not entirely distinct. There are inevitably some regularities across contexts that could be understood to guide research and practice regarding the pressures of agile translation over time. Yet there is no general way to conceive of these differences to show how different institutional contexts shape the translation of agile. Institutional theory suggests that organizations will align practices with their institutional imperatives (DiMaggio and Powell 1983), so exploring this alignment in agile methods is crucial. Thus, in this research, we seek to understand how *different institutional contexts shape the translation of agile practices over time.* In particular, we look for general regularities that may explain this translation process.

We conducted an exploratory study of organizations from four different institutional contexts—one entrepreneurial software company and three more traditional organizations (i.e., manufacturing, government, and professional service). We draw on translation theory, which examines how practices change as they are diffused across different institutional settings (Czarniawska and Sevón 1996, Pries-Heje and Baskerville 2017), to explore the institutional logics that shape development processes. We identify two theoretical constructs as the regularities across logics that appear to influence the translation process: *pluralism* and *dynamism*. These concepts enable us to move beyond specific institutional logics to theorize about regularities across diverse contexts. Additionally, we identify a "ratcheting" process, which describes how the translation of agile gradually shifts over time to better align with a relevant institutional logic. The notion of ratcheting emphasizes that translation occurs in a punctuated, stepwise manner, where each adaptation builds off the previous adaptation. Organizations make incremental changes to a limited set of agile practices, which then cascade into further changes over time. This ratcheting process contrasts with research on agile tailoring, which generally assumes discrete decisions aimed at optimizing fit.

We theorize that cases characterized by high dynamism but low pluralism (i.e., entrepreneurial settings) tend to embrace agile practices. In contrast, cases with higher pluralism, lower dynamism, or both higher pluralism and lower dynamism tend to move away from canonical agile practices. Pluralistic contexts ratchet toward documentation, whereas less dynamic contexts favor slower temporal pacing. These findings have implications for both research and practice, especially as agile methods expand beyond software development (Bianchi et al. 2022, Reineke et al. 2025).

The Institutional Practice of Software Development

Software development methods offer legitimate practice scripts for developers (Avgerou 2000), and their organizations maintain legitimacy by adhering to these methods' practices (Vial and Rivard 2016). Thus, software development methods are an example of institutional practice (Avison and Fitzgerald 2003), entailing specific values, assumptions, symbol systems, and norms (Avgerou 2000, Qiu et al. 2017). Agile refers to a family of software development methods, such as Scrum or Extreme Programming (XP), that consist of normative practices[1] (e.g., daily stand-ups, pair programming, etc.; see Sidky et al. 2007, Maruping et al. 2009, Diebold and Dahlem 2014) intended to help organizations innovate quickly and flexibly (Recker et al. 2017, Kum et al. 2024).

Agile methods emerged in the early 2000s, emphasizing iterative development as a response to the rigid, phased waterfall model (Fowler and Highsmith 2001, Schwaber and Beedle 2002, Beck and Andres 2004, Cockburn 2004). From prior literature (Conboy 2009, Vidgen and Wang 2009), we identify four general dimensions that differentiate traditional from agile approaches (Table 1):[2] (1) documentation (heavy versus minimal), (2) temporal pattern (long phases versus short cycles), (3) coordination mode (formal distributed teams versus collocated collaboration), and (4) governance (manager driven versus peer/architecture driven).

Agile *documentation practices* emphasize face-to-face interaction and natural language communication (Hummel et al. 2015), reflecting a reduced focus on the formal documentation common in structured development (Cockburn 2002). Documentation is less central in agile settings, because changes are implemented and tested frequently (Dybå and Dingsøyr 2009).

Temporal patterns are the ways in which work activities are organized over time. Structured development emphasizes sequential, temporally extended phases with strict specialization of effort. Agile practices entail iterative, incremental development, with short cycles in which all activities are pursued (Larman 2003, Conboy 2009). Iterative development is central to an agile team's ability to sense and rapidly adapt to changing requirements (Lee and Xia 2010).

Table 1. Dimensions of Structured vs. Agile Approaches

Dimensions	Structured approaches	Agile approaches	References
Documentation practices	Documentation-centric	Minimal documentation	Cockburn (2002), Pikkarainen et al. (2008), Hummel et al. (2015)
Temporal pattern	Phased: longer durations, parts of process	Iterative: short bursts, entire process	Beck et al. (2001), Larman (2003), Dybå and Dingsøyr (2009)
Coordination mode	Collocated or distributed; generalists	Primarily collocated; specialized	Cockburn (2002), Hummel et al. (2013), Hummel et al. (2015), Kudaravalli et al. (2017)
Governance (management and roles/architecture)	Hierarchical—management directed	Egalitarian—architecture directed	Maruping et al. (2009), Vidgen and Wang (2009), Abrahamsson et al. (2010), Lee and Xia (2010)

Coordination describes how teams organize activity. Agile teams are intended to be collocated and coordinate through informal and tacit mechanisms (Cockburn 2002, Kudaravalli et al. 2017). Eschewing strict specialization, agile developers must perform a wide range of tasks and attain general skills (Schwaber and Beedle 2002, Hummel et al. 2015).

Governance involves the structuring of the development process through software architecture. In contrast to structured development's central planning and control, agile approaches promote decentralized, self-organized decision making (Vidgen and Wang 2009, Hoda et al. 2012). This is, in part, enabled by modular architectures that substitute for some interaction (Abrahamsson et al. 2010).

Agile Tailoring

Organizations tailor agile methods to address their needs (Avison and Fitzgerald 2003). There are two general approaches to research on tailoring: *methods engineering* and *contingency factors* (Fitzgerald et al. 2006). In methods engineering, teams formulate normative guidelines to design specific methods in use (Brinkkemper 1996, Henderson-Sellers et al. 2014). Contingency factor approaches emphasize characteristics of the development context to select appropriate methods (Iivari 1989, Avison and Wood-Harper 1991). Both approaches are prescriptive (Fitzgerald et al. 2006).

In research on agile tailoring (see Online Appendix A for a review), there is a notable absence of adaptation theories. Existing research characterizes tailoring as a decision and subsequent enactment. No studies theorize about abstracted regularities of organizational contexts to examine *why* specific factors trigger unique changes, nor do they probe the underlying mechanisms of *how* agile tailoring unfolds over time. Instead, existing research paints a picture of discrete choice—this decision-centrism is a core assumption of both the methods engineering and contingency factor approaches (Conboy and Fitzgerald 2010). However, some detailed accounts reveal an emergent process of experimentation and tinkering, undetermined factors, preferences, organizational contexts, and power dynamics (e.g., Conboy and Fitzgerald 2010, Wang et al. 2012, Cram and Newell 2016). Pries-Heje and Baskerville (2017) describe this contextually shaped emergence of agile practices in terms of "translation."

Translation and Agile Development

Translation theory (i.e., Scandinavian tradition of institutional theory; Wæraas and Nielsen 2016) explains how practices are adapted across institutional contexts over time. Whereas much of institutional theory explores how isomorphic practices diffuse (i.e., DiMaggio and Powell 1983), translation theory focuses on the construction of institutions in local contexts (Czarniawska and Mazza 2003). Practices (like agile) travel as ideas from organization to organization, and each "translates" the practice differently to fit local imperatives (Czarniawska and Sevón 1996).

Pries-Heje and Baskerville (2017) drew upon translation theory to explore the social process of agile software development as it is enacted, or articulated, in practice. Translating agile involves drawing on different agile tools and techniques and articulating them in a working method that fits the needs of a local context. A key premise is that agile practices are likely to be translated in similar ways within similar institutional contexts (Morris and Lancaster 2006). The notion of *institutional logics* can help us to analyze and characterize similarities that emerge. Institutional logics are "rationalities" associated with an institution, including the symbol systems, interpretive schemes, norms, and organizing principles of that institution (Friedland and Alford 1991). Institutional logics are used to show how locally situated rationalities are embedded in broader institutions (Thornton and Ocasio 2008, Berente and Yoo 2012) and how organizations translate technologies

in practice in ways that are shaped by regularities in their institutional environments (Nielsen et al. 2014).

Organizational actors draw on established institutional logics to guide their actions. When technologies and practices are introduced that draw on different institutional logics, this can lead to tensions because the new practices require fundamental change to the goals and assumptions of that domain (Berente and Yoo 2012, Hansen and Baroody 2020). Organizational actors can respond to these tensions in a variety of ways: resisting or rejecting the change, decoupling their activity while appearing to comply with the change, or attempting to co-opt the new technology or practice (Berente et al. 2019). Of course, they can also comply with and adapt the new technologies and practices, doing something fundamentally different from before, with implications for organizational identities, power structures, and meaning (Kellogg 2022). Research shows that the adaptation of agile can change the institutional logics that guide development practices—agile is expected to increase flexibility and innovativeness, but may also stifle innovation and foster short-term thinking (Annosi et al. 2022). Such research starts with agile adaptation and examines its outcomes, but does not explore how this adaptation is shaped over time. Research on agile tailoring suggests that different contexts should implement agile differently. However, tailoring research typically reflects a decision-oriented focus and does not theorize about the temporal adaptation of agile. Thus, we adopt a translation lens on agile development to explore how institutional logics shape the enactment of agile, focusing on regularities across logics rather than any particular institutional logic, thereby zooming in on dimensions of logics that appear to influence how agile is translated. We construct a general, institutional explanation for why and how organizations tailor agile in the ways that they do over time.

Research Method

We conducted an inductive, qualitative field study, following a general approach to grounded theory methodology (Strauss and Corbin 1998, Birks et al. 2013). Our study design was a multicase exploration, with four cases theoretically sampled (Strauss and Corbin 1998; see Online Appendix B for details) to maximize contrast in institutional contexts (Table 2). We started with two organizations—a tech startup (StartUp) and an information technology (IT) consulting firm (Consult)—and inferred that context differences appeared to significantly influence agile adaptations. We then added a government agency (Infra) and a manufacturing firm (Auto) sampling on differences to include contexts with contrasting institutional logics (public sector, industrial).

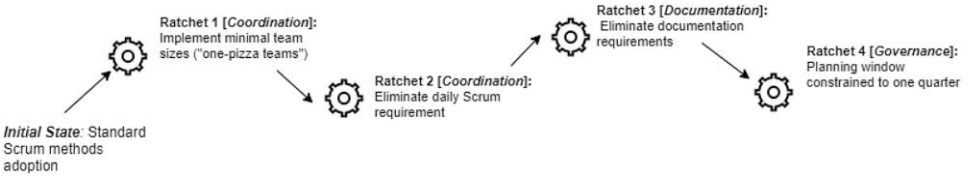
To analyze the data, we performed iterative open coding of interview transcripts and documents, identifying incidents of changing agile practices. Through constant comparison, we grouped codes into higher-level concepts and referred to relevant literature to refine our categories. This process was consistent with the Gioia et al. (2013) methodology for constructing a data structure—first-order descriptive codes aggregate into second-order themes and overarching dimensions (see Online Appendix B). The analysis revealed four main dimensions of agile practice evolution—documentation, temporality, coordination, and governance—and two salient dimensions of institutional contexts that distinguished the cases, dynamism and pluralism. The constructs of dynamism and pluralism are second-order constructs that emerged from comparing dimensions of the logics (first-order constructs; see Online Appendix B).

To understand the agile translation process over time, we traced the sequence of agile adaptations within each case. We identified distinct "ratchet" events in each organization—punctuated changes in practices—and ordered them chronologically using diagrams as a sort of memo (Birks et al. 2013). According to Birks et al. (2013, p. 2), "[m]emos can take the form of diagrams, text narratives, propositions, mind maps, and other techniques that are suitable to both the idea of being documented and the cognitive preferences of the researcher as sense maker." We used temporal diagrams (Figures 1–4) in conjunction with informant narratives to construct our view of ratcheting over time, with an initial change (first ratchet) followed by subsequent modifications (following ratchets). This approach was a creative form of temporal bracketing (Langley 1999) that allowed us to construct a process sequence to examine how one change led to another within the same case. We compared these patterns across cases for

Table 2. Overview of Cases

	Cases			
Descriptive dimensions	StartUp	Consult	Infra	Auto
Institutional context	Technology start-up	Service organization	Government agency	Industrial manufacturer
Domain	Social media aggregation /analytics	Software consultancy for financial services	Government computing infrastructure	In-house automotive software applications
Sampling criteria	More dynamic, less pluralistic	More dynamic, more pluralistic	Less dynamic, more pluralistic	Less dynamic, less pluralistic

Figure 1. Agile Ratcheting Articulation at StartUp



more general insights. Our inductive analysis yielded a set of theoretical propositions that link contextual conditions to changes in agile practices.

Findings: Dynamism and Pluralism

Dynamism refers to the degree of volatility in the development context, whereas pluralism denotes the inclusion of diverse stakeholders in the development process. It is important to note that dynamism and pluralism are our interpretations of the development team context and are not intended to summarize a given organization's context overall. In other words, an organization may be quite dynamic or pluralistic, but an agile development team within it may work on stable, relatively nondynamic software and interact with a limited set of perspectives.

Dynamism

StartUp's development teams operate in a highly competitive context, where customer expectations of solutions are in continuous flux. To address this rapid pace of change, the firm emphasizes an iterative approach to strategic planning and software development. StartUp's processes reflect nested cycles of planning that enable progressive elaboration of market foci—moving from firm-wide "strategic arcs" established annually to quarterly goals and, finally, individual sprint plans. StartUp's managers plan for only one quarter at a time because they expect conditions to change:

> What we know is the yearly strategic arc and we know what we are going to do in Q1 [quarter 1]. "So okay, what do we do in Q1 to meet that arc?" Right? So it's conceptualize it, and then we replan every quarter to see what we achieved and what adjustments need to be made. [1]

Consult also operates in a relatively dynamic context. The firm has grown rapidly since its founding in the 1990s. The company's customers require that they include wave upon wave of new software functionality. The need to reconcile the demands of diverse customers into a single platform contributes to the tumultuous environment:

> [The customers] have different requirements regarding the calculation kernels in the different products that we have to offer and have to calculate. … [So we] split out releases every two weeks because there were different customers always washing in new requirements. [20]

The environment of Infra is comparatively less dynamic. Enhancements to the scheduling system were mostly incremental, with no strict deadlines. The project moves at a steady pace, and there are few risks. One developer characterized the pace of the development as "chugging along" ([7]).

Auto's software development context was also relatively less dynamic. In the project we studied, the IT strategy team follows a five-year road map with periodic project assessment and evaluation. Auto's software development followed the pace of their vehicle production:

> The way it works is … when your vehicle lines are five, six, seven years old … the top auto companies turn their product every two, three years, four years, completely, almost. [12]

Pluralism

StartUp's development environment is relatively dynamic, but its stakeholder pluralism is comparatively low. StartUp informants consistently evidenced a homogenous entrepreneurial mindset. This consistent perspective underlies StartUp's approach to coordinating work activities, which involves small, collocated teams and iteratively developing and testing innovative projects.

Figure 2. Agile Ratcheting Articulation at Consult

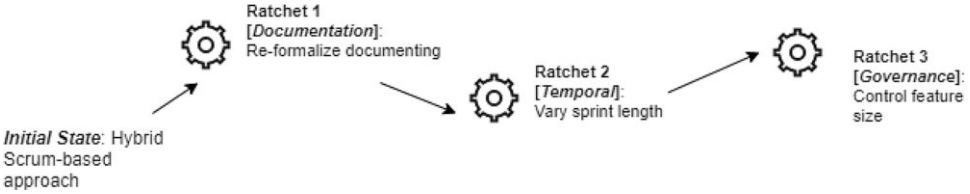
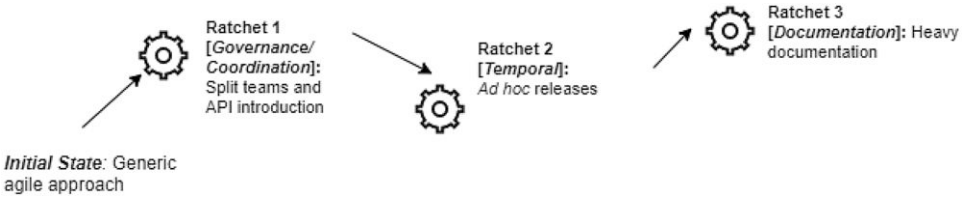


Figure 3. Agile Ratcheting Articulation at Infra



Consult, however, was more pluralistic and the development environment spanned stakeholder perspectives across a variety of financial industries (e.g., banks and insurers), with diverse institutional backgrounds (e.g., private banking, public banking, insurance, etc.), each regulated differently:

> Firstly, we have rather a lot of clients … they all have similar projects but they are never the same. You know, it's like with SAP, we create a bundle of standardized software but we always have to customize it quite a lot, quite intensively as well. [19]

Infra's development environment is low in dynamism, but it is quite high in pluralism. Infra is a virtual organization spanning a dozen universities and various scientific fields. The software is developed for scientists from different disciplines with a variety of scientific workflows and conflicting demands. As a developer explained:

> People wanted diametrically opposed things. I mean if you talk to a group of users, they're going to want two different things. So it can't really be a requirements list because not everybody could get everything that they wanted. [9]

The institutional context for development at Auto was not particularly pluralistic. Auto is an industrial car manufacturer with a relatively stable environment that is deeply established. Software development reflects the engineering approach that dominates the organization. As a manager indicated, "[There are] a lot of similarities with how we build cars and how we build software" ([11]).

Findings: Ratcheting and the Translation of Agile Practice

In each of our cases, the agile method applied underwent a series of incremental modifications—a stepwise translation process we term ratcheting. A ratchet is a single, punctuated change in an agile practice, after which the new practice state "locks in" in that it forms the basis for subsequent change. These changes were interdependent: each adjustment set the stage for further adjustments. Over time, one ratchet led to another, gradually moving the team further from the initially implemented practice. For example, in Consult, the team's first ratchet was to extend sprint lengths from two to four weeks to accommodate more documentation and approval time for diverse clients. This, in turn, slowed feedback cycles, which eventually prompted formalizing additional up-front planning (a subsequent ratchet). In other words, introducing heavier documentation tightened one screw (documentation) and loosened another (tempo) in the agile process, illustrating how one ratchet triggered the next. Ratcheting unfolds as a chain of changes. In the following, we detail prominent ratchets that we observed in our cases.

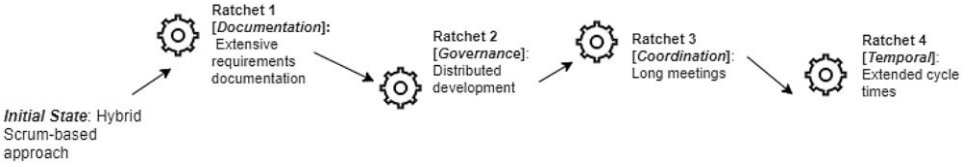
StartUp

StartUp employs a lightweight approach to documentation, intensive collaboration, autonomy for individuals, and the development of broad-based skills. The firm largely adheres to a Scrum methodology, with some adaptation. A consistent theme within the StartUp environment is doing only what makes sense for a given task or project. According to StartUp managers:

> [W]e are doing more lightweight requirements documentation, because the primary purpose of that is really just to connect the product guys with the engineering guys. … As long as they have a common language for understanding, it serves its purpose. [1]

Although they espouse a Scrum approach, StartUp regularly implemented new guidelines to "just do what makes sense" and continually and incrementally adopted increasingly agile practices.

Figure 4. Agile Ratcheting Articulation at Auto



Ratchet 1: Minimizing Team Size. Early in the implementation of agile, StartUp found large teams unwieldy. They determined that the additional communication to integrate components was unnecessary because teams were consistent in their understanding of goals (low pluralism), and communication for communication's sake got in the way of the rapid decision cycles (high dynamism) they needed to meet to keep their modular products updated:

> That's why we like to keep these teams very small because then the communication back ends up being smaller and shorter and extremely fast. [1]

They created the norm of "one pizza teams" (i.e., teams that could be fed with one pizza; only two or three individuals) to minimize coordination challenges and ensure quick adaptation.

Ratchet 2: Eliminate Daily Scrum. StartUp also found that, with very small teams and with high alignment (low pluralism) of the developers, they did not need as many formal check-ins. They eliminated daily stand-up meetings, a hallmark of many Scrum projects:

> We do not do daily stand-ups … two people working together, they know what's going on and it's really old hat to have 25 people stand around talking about what they did every day. [2]

Ratchet 3: Reducing Documentation. Flowing from smaller teams and modular products, StartUp teams found requirements and other project documentation to be superfluous. Not only was there minimal need because development teams were largely on the same page (low pluralism), but the documentation was too quickly outdated to be useful (high dynamism):

> The sprint documentation like requirements are very lightweight … the purpose of these agile methods is to actually do stuff, not really write stuff, for that quickly goes out of date anyway. [1]

Ratchet 4: Quarterly Planning. Initially, StartUp engaged in road mapping for software products for one year or more. However, managers observed that changes in the marketplace rendered much of this planning obsolete. So the firm switched to annual determination of broad strategic objectives ("strategic arcs"), with detailed planning limited in most cases to no more than one quarter in advance. Thus, StartUp essentially eliminated long-term planning for their software products:

> Each quarter we plan out what would be the next quarter. We very rarely plan out two, three quarters in advance. … We re-plan every quarter to see what we achieved and what adjustments need to be made. … It's very dynamic and very agile. [1]

Overall, StartUp enacted agile development into a highly agile set of practices. When StartUp departed from Scrum, it was to move in a *more* agile direction (Figure 1).

Consult

Agile development at Consult was implemented to flexibly adapt to customer needs:

> The best feedback is when you show the features to the customer try to get a first feedback and try to fix some things when the customer says: "Oh that was not so good I would like to have it more like this and like this." [21]

Consult nevertheless looked to retain extensive documentation of client requirements for contracting. They balanced this with an agile, iterative approach that involved intensive interaction with the clients. Several practices of different agile methods (e.g., daily stand-up meetings, continuous integration, feature-driven development) were adopted and combined with more formal, plan-driven practices (e.g., modeling requirements by use cases in written form, quality gates).

Although Consult's context of high dynamism resembles StartUp's, the firm's translation of agile practices is markedly different. Most notably, we see a greater emphasis on formalism and the need for multiple coordination mechanisms between developers and clients. Thus, although Consult has achieved a more agile development approach, the enactment of agile in the face of high pluralism necessitated the reintroduction of substantial formalism in documentation, coordination, and governance.

Ratchet 1: Reformalize Documentation. Before implementing agile, Consult separated design and implementation groups, coordinating their activity through requirements (e.g., specifications, use-case diagrams) and other project documentation. When initially implementing agile, Consult brought the two groups together, hoping to eliminate much of the documentation in line with the agile philosophy. However, they found that they needed to detail the perspectives of the clients carefully and to align development with other development teams. A manager explained:

> So we are going back to having designs more concretely [documented] and really saying what details we want to have from the requirement side. … We tried to do [both] the analysis and the requirement definitions in this kind of agile style, and this didn't work for us. [19]

Consult reintroduced much of the requirements documentation that enabled them to manage customer

expectations and the multiple perspectives (i.e., pluralism) of their environment.

Ratchet 2: Vary Sprint Length. Although Consult initially attempted to implement sprints every two weeks, they realized that with greater documentation, sometimes they needed to extend this in order to get client and management approval, which required time. This documentation time was required then in every sprint, which encouraged them to extend sprint durations:

> We decided for the next iteration that we take more time … the beginning of such an iteration are always the same. … So we are more going back to three or four weeks. [18]

Ratchet 3: Control Feature Size. Sprint sizes were extended to up to 15 days and the coordination among stakeholders required a large portion of the sprint duration, so Consult reduce the scope of new development activity to roughly 50% of the sprint. Features expected for sprints were scoped down:

> So the iteration is longer, but the features are the same … normally a feature development takes about one or two weeks and fits in the iteration. Sometimes three weeks, when you have some more bugs or the feature is more complex than you expected at the beginning. … [18]

Consult also evidenced a series of ratchets in their translation of agile development (Figure 2). Unlike StartUp, however, Consult struggled to discern the appropriate degree of agility they wanted to pursue. Consult balanced the combination of high dynamism and high pluralism in their professional service context with pursuing innovative solutions and demands for coordination between stakeholders.

Infra

Infra's software development processes reflect a hierarchical logic. Multiyear funding for the scheduling software came from a government source and the project was led by an infrastructure center in Illinois. The goal was to develop the system with a modular architecture and extending features through an agile process with intensive use case development with stakeholders.

Ratchet 1: Split Teams. Infra development was driven by a small team in Illinois, but after the increase in funding, there was pressure from the funding agency to spread activity across states to meet the pluralistic interests of government, which split the project across Texas and Illinois. Texas led the user interface, and Illinois led the database application development. This split in functionality necessitated more coordination, leading to the development of an application programming interface (API) to align efforts:

> We're writing against that [API] and so implementing the API completely as well as writing full test coverage for that API for software engineering, which gives us the advantage of that API is tested from both sides. [7]

Ratchet 2: Ad Hoc Releases. The distributed teams had weekly meetings initially to coordinate the splitting of the application and development of the API, and this meeting was continued to prioritize requirements and update each other. The intent was to one day fall into a pattern of regular sprint-based release cycles, but this never emerged. There was little time pressure on the project, so limited impetus for scheduled releases. Therefore, the team followed an ad hoc approach to development and releases came only when well-tested, much like the traditional software approach: "When stuff changes, like tickets get posted … 'This is not working,' okay we'll fix it and try it again" ([7]).

Ratchet 3: Heavy Documentation. Initially, the two teams worked from an informal list of requirements compiled by the Illinois team. Over time, however, the funding agency, through annual reviews, wanted to see evidence of a more disciplined approach to the project overall. The project had multiple competing demands from a variety of users across scientific disciplines, and they needed to instill a significant amount of process formalization overall, and this influenced the scheduling project, which ratcheted the process to greater formalization (Figure 3):

> Formalized means that before this the requirements were being tracked by a bunch of different people in their own documents on their desktops and that kind of thing. We've moved all of this into a design document now. It has use cases and a mapping of the requirements to the use cases. [6]

Auto

Auto's software development approach was traditionally a structured process with abundant documentation, much like automotive product development. The bill of materials software project was an opportunity to experiment with agile. They put all developers for the project in a single room, encouraged pair programming, test-first development, continuous communication, smaller units to iterate, and an expectation of faster, leaner cycles of software production. However, Auto's translation of agile ended up involving more documentation than planned, and also discrete phases for approvals that involved "freezing" requirements with the business unit. Auto translated the agile process in the direction of a traditional structured process

over time, in the context of a relatively less dynamic context and in the direction of the dominant, nonpluralistic way of doing things in the organization. The result involved more documentation and longer cycle times.

Ratchet 1: Extensive Documentation. Over time, requirements from managers and engineers necessitated the same elaborate set of documentation that was required for all projects, including project planning, specifications, process and architectural modeling, and full approval documentation. This was required because of established ways of dealing with projects in the organization, along with relatively little time pressure—software projects were treated like any engineering project:

> Usually when the requirements get locked and dumped into an SDM [software development methodology] artifact … there's another boiler plate that we fill out and we make them agree to it. … And then that gets captured at a point in time, so that when we need to go back and say, "Well, you guys approved this content on this date," just in case there's a disagreement. [11]

Ratchet 2: Introduce Distributed Development. Auto initially collocated their agile project team to foster informal coordination. However, across all software projects, management chose to offshore much of the development to India, signaling that cost cutting was valued over innovativeness in internal software projects. The agile project was no exception. This disrupted the collocation of the team:

> We're really struggling to make it work because our development teams are in India and so we can't really be agile … the decision to create an IT organization in India was part of what our current VP told the Board that we were doing as part of our cost cutting. [13]

Ratchet 3: Increase in Meeting Duration. Coordination became more cumbersome, and the outsourced part of the process introduced more iterations and checkpoints for the core team. One developer indicated that it takes time to catch up during testing phases that required daily three-hour meetings: "Every morning six to nine. Six to nine, three hours." [16]

Ratchet 4: Extended Cycle Times. Because of increased coordination with India and extensive documentation and approvals, Auto achieved fewer iterations with extended time periods. The desire to iterate ran against company conventions that required a good deal of formality, planning, and discipline. The bill of materials software team attempted to take smaller portions of the projects at a higher cycle time for them, but this involved months of calendar time. This ratcheting process resulted in reduced cycle times (Figure 4):

> We structured it a little bit more, but we still kept it very agile. We still were doing fairly rapid releases, every couple of months. For the first four or five months we did two or three releases … then it slowed down. [11]

Discussion

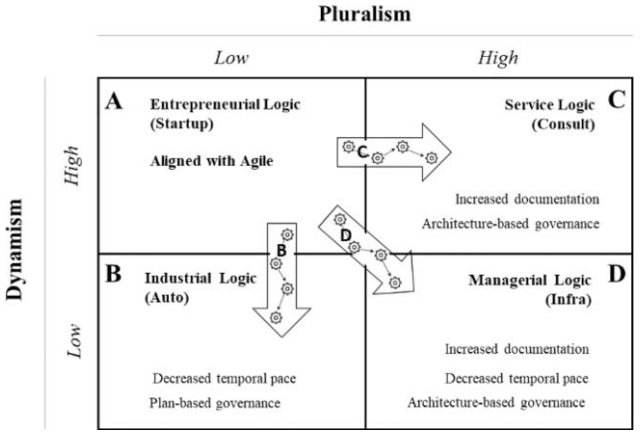
We found that agile teams in different institutional contexts translated agile practices in ways that relate to the dynamism and pluralism of their development environments. The translation of agile practices did not occur solely through discrete decisions at the time of adoption in any of the organizations, as implied by much of the research on tailoring. In contrast, each organization initially implemented a relatively canonical agile development approach, but later translated the method differently over time. The main contribution of our work is to theorize *how* agile practices translate across different institutional contexts. Organizations are known to tailor agile practices to fit their local contexts (Iivari and Iivari 2011, Campanelli et al. 2018).

We characterize organizational contexts in terms of broader institutional characteristics of dynamism and pluralism, which are general dimensions related to institutional logics that appear to influence agile translation. We then relate these characteristics to the agile practices we identified (documentation, cycle times, coordination, governance). All of the organizations began with relatively canonical agile practices, but three of them ratcheted away from canonical agile in different directions (Figure 5). StartUp was the exception; its practices remained aligned with agile—indeed, becoming even more agile over time (cell A). Auto, which was relatively low in dynamism and low in pluralism decreased its temporal pace and ratcheted up plan-based governance (cell B). Consult, which was relatively high in pluralism and high in dynamism, ratcheted toward increased documentation and architecture-based governance (cell C). Infra, which was relatively high in pluralism but low in dynamism, ratcheted toward increased documentation, decreased temporal pace, and architecture-based governance (cell D). Next, we theorize about what this means for each set of agile practices.

Documentation Practices

Two of our cases showed increased documentation over time—Consult and Infra (C and D in Figure 5). Consult initially attempted to use light documentation, but pressures from their diverse customer demands required them to ratchet up documentation. Infra, as a

Figure 5. Patterns of Agile Translation



Notes. Cell A is the starting point of canonical agile practice, with high dynamism and low pluralism. Other cells represent directions for agile translation. Arrows are movement in that direction. Cells B and C represent an "or" logic in the direction of agile translation; B is low dynamism, and C is high pluralism. Cell D indicates an "and" logic of both high pluralism and low dynamism. The arrows represent the ratcheting process from canonical agile toward the different cells.

government project, utilized more extensive documentation over time to track requirements and report on the project. In both cases, due to the pluralism of the institutional contexts, the organizations ratcheted to more documentation over time (arrows C and D in Figure 5). Based on these observations, we propose the following.

Proposition 1. *High pluralism in the development environment will be associated with increasing levels of documentation in the translation of agile development.*

In pluralistic contexts, documentation is used to navigate the diverse conceptions of the software being developed. Developers try to create common ground (Clark 1996) and shared understanding (Corvera Charaf et al. 2013) of complex situations—seeking to pin down abstract concepts in contexts marked by plurality. Research shows how documentation helps organizations to navigate diverse customer requirements (Czarniawska and Mazza 2003), which explains the connection between pluralism and documentation. This was especially apparent in Consult and Infra. In the case of Auto, which is a less pluralistic organization, the development group still needed to pin down the line-of-business customers to a specific set of requirements from the very start of our data collection.

Temporal Patterns

Another observation involves the relationship between temporal patterns and dynamism. Canonical agile involves short cycle times, daily standup meetings, daily refactoring, and quick sprints. Organizations facing high levels of dynamism, such as StartUp (three-week sprints) and Consult (two- to four-week sprints), reflected much greater adherence to this short-cycle ideal than Infra and Auto, each of which spent months per iteration. Infra and Auto faced minimal urgency and pressure for changes from stakeholders. Infra's government project had no hard deadlines and experienced only incremental requirements changes, allowing the team to stretch out iterations over months. Auto's IT unit aligned software releases with the automotive product cycle (multiyear), reflecting a deliberate, slower cadence. With fewer external triggers for rapid change, these teams could afford a slower, more planned pace. Indeed, although Infra and Auto sped up their processes somewhat during agile adoption, they "chugged along" (as one developer put it) at a much slower rhythm than StartUp or Consult over time. This observation leads to our second proposition.

Proposition 2. *Low dynamism in the development environment will be associated with longer iteration cycle times in the translation of agile development.*

In high-dynamism cases (StartUp, Consult), there was constant pressure to deliver features quickly—fast cycles were a competitive necessity (Anupindi et al. 2006, Giardino et al. 2016). Thus, low-dynamism environments tend to have longer cycles because context demands do not require speed.

Coordination Practices

Canonical agile involves collocated development, but in our cases, there was significant distributed, noncollocated work, and this appeared to be associated with either low dynamism, high pluralism, or both. Consistent with agile principles, fast cycle times are enabled, in part, by collocation. StartUp was entirely collocated. Consult's developers were collocated, although client partners, consultants, and some managers were not at the same location as the development team. In both organizations, the dynamism was enabled by collocation, and in Consult, any noncollocated activity was associated with greater pluralism. In Infra and Auto, distributed development was driven by managers' decisions, without regard for agile goals. Infra was not under as much time pressure (dynamism) and was required by funding agencies to distribute activities (pluralism). Similarly, Auto went overseas to save costs, and this introduced diverse, international stakeholders in the development process (pluralism). Apparently, time demands were less important than cost issues (lower dynamism). It is not that dynamism or pluralism necessarily caused the distributed work, but they were associated with that form of coordination, leading to our proposition.

Proposition 3. *Low dynamism and/or high pluralism in the development environment will be associated with*

increasing distribution of coordination in the translation of agile development.

These points are consistent with previous observations that contexts low in dynamism will move away from group-based team coordination (Gregory et al. 2013). This adaptation resulted in additional control points in the process, with developers from different locations participating in testing and in formulating instructions. Infra and Auto (cells B and D in Figure 5) were the most distributed, and distributed teams are expected to increase documentation and move away from certain agile practices (Fitzgerald et al. 2006, Bass 2016). This is the case in virtual teams (Moe et al. 2016) and larger programs (Gustavsson 2021). It is difficult for distributed teams to maintain agile processes (Sarker and Sarker 2009). Being in close proximity will speed up team communication and coordination.

Governance Practices

In terms of governance, each project employed active, professional managerial practices for software governance. However, Auto was the only context in our sample that emphasized a planned approach, with governance decisions following a functionality road map. The others emphasized architecture in varying degrees. In the case of StartUp, requirements had a temporary quality, and the organization prepared for indeterminacy by focusing on architectural compatibility. Architecture, in terms of strict adherence to modular interfaces, enables the team to evolve the software in unforeseen, generative ways according to standard interfaces (Yoo et al. 2010), while reusing components and avoiding overspecification and the accumulation of technical debt (Rolland et al. 2018, Hahn et al. 2025). Consult emphasized a single vision for the platform's architecture, which provided constraints and interfaces for customer requirements. Infra was less concerned about open-endedness and dynamism, instead using architecture to reconcile multiple distributed development processes. In this sense, the modular architecture fostered the decentralized governance that enables distributed agility (Tiwana and Konsynski 2010). In sum, Infra adopted an architecture-driven approach to manage pluralism, StartUp for dynamism, Consult had both, and Auto had neither.

Proposition 4. *High dynamism* and/or *high pluralism in the development environment will be associated with architecture-driven governance in the translation of agile development.*

Taken together, these observations indicate that development contexts high in dynamism but low in pluralism are most likely to adopt agile practices faithfully. In our sample, StartUp (cell A) was entirely aligned with all dimensions of agile—small, collocated teams enable consistent communication and a fast pace, and minimal documentation. Among our cases, StartUp was the only organization that employed minimal documentation and strict collocation. StartUp operates in a dynamic institutional environment that is comparatively less pluralistic—the innovative "entrepreneurial" logic is prevalent. Their activity is consistent with research that indicates that startups are particularly inclined to tailor software development practices to reduce documentation (Coleman and O'Connor 2008) and also that startups are apt to apply agile methods that increase speed (e.g., by elimination of daily standups; see Pantiuchina et al. 2017). StartUp avoided long-term planning, implying an effectuation approach to strategy (Sarasvathy 2001). Effectuation refers to the emergent, experimental, "lean" approach to entrepreneurship that is typically contrasted with plan-based strategy (Ries 2011). This effectuation perspective lies at the root of agile implementations that are even leaner than is typical (Poppendieck and Cusumano 2012). This leads to our final proposition.

Proposition 5. *High dynamism and low pluralism in the development environment will be associated with canonical translation of agile development.*

As we move to more pluralistic and/or less dynamic institutional contexts, agile practices are tailored to move away from canonical agile practice. This observation is consistent with expectations from the literature (Cao et al. 2009, Iivari and Iivari 2011, Cram and Newell 2016). However, we extend this research, by observing that each of the more traditional contexts translated agile development differently. We offer the theoretical foundation of middle-range[3] core categories of dynamism and pluralism as critical axes for understanding how these development contexts differ. Further, we draw on accounts of actual agile translation, thus rooting our argument in empirical findings that corroborate and elaborate on existing prescriptive research. The main contribution of our work is to theorize *how* agile practices are translated across different institutional contexts. A key advantage of doing so and applying an institutional perspective is situating idiosyncratic practices within broader social regularities (Berente and Yoo 2012), and in this research, we identify specific institutional characteristics and relate them to agile translation (Pries-Heje and Baskerville 2017).

The Ratcheting Process

In addition to treating each agile practice independently, it is important to note how they relate to one another. This is the key to the stepwise ratcheting practice we identify. By using the term "ratchet," we emphasize the accumulation of changes—how one change, in turn, influences a subsequent change. For example, we observe the interconnectedness of documentation and temporal practices, revealed most vividly in the Consult

case. As they recognized the need for greater documentation to understand customers, create common ground, and coordinate between diverse stakeholders, this impacted sprint times directly by necessitating longer and more variable cycle lengths in line with demands for creation, review, negotiation, and approval of the documentation. This involved "tightening" the metaphorical set screw of documentation for pluralism, and "loosening" the corresponding set screw for temporality.

Similarly, we observe a relationship between the governance of development efforts and the modes of coordination employed. Governance decisions influence interactions within teams, and different modes of interaction necessitate certain approaches to governance. Infra team members were not collocated, limiting their interaction. To address this coordination impediment, they orchestrated much of their activity through a modular interface and project governance approach emphasizing architecture. This did not eliminate the need for interaction—indeed, it appeared to necessitate intensive meetings, which were also evident in both Auto and Consult.

Our notion of ratcheting involves incremental, stepwise adjustments that are in many ways consistent with punctuated adoption long identified in organizational change models (Tyre and Orlikowski 1994, Lyytinen and Newman 2008). It is not gradual evolution, but instead a step at a time—punctuations in the process. However, to this, we add the accumulative aspect of the change, where each step moves in a direction toward or away from canonical agile practice that influences subsequent changes. The tightening and loosening of agile comes to the foreground with the ratcheting metaphor. This is reminiscent of a cascading view of innovations (Boland et al. 2007), whereby one change results in a subsequent change but adds the accumulative and directional (i.e., tightening and loosening) aspect in terms of more or less canonical agile practices. Such a view is consistent with recent perspectives on translation ecosystems (Nielsen et al. 2022, Elmholdt et al. 2025) that characterize the translation process in terms of the variety of roles that interact with each other and reciprocally shape the enactment of processes such as agile over time.

Contribution

How do agile methods evolve in practice? Our findings suggest that it is largely due to organizations adjusting to institutional demands: high pluralism demands alignment through documentation and coordination, whereas low dynamism breeds complacency toward speed and encourages formal control, among other factors. These contextual forces drive the incremental adjustments (ratchets) that cumulatively reshape agile implementations. The notion of translation is a way to think about the diffusion of practices, where they change as they are appropriated differently in different situations (Nielsen et al. 2014, Pries-Heje and Baskerville 2017). In his seminal work on agile development, Cockburn (2002) argues that agile practices should be tailored to the specific application. Mission-critical, complex applications should reflect a more heavyweight approach with greater formality, documentation, and ceremony, whereas less complex applications should involve lighter-weight adaptations. Such tailoring has been widely observed in industrial practice (Fitzgerald et al. 2006, Campanelli et al. 2018). The way organizations implement agile methods should fit their strategy and culture (Iivari and Iivari 2011, Ramesh et al. 2019). However, the existing literature provides little explanation for why certain agile adoption patterns occur in different institutional contexts, beyond the fact that they do. Nor does research describe how tailoring occurs over time, beyond merely observing that it happens. Our results extend prior tailoring studies by moving beyond cataloging tailoring choices to theorize the influence of environmental context on those choices. For instance, whereas prior work might note that government projects often involve more complex processes, or traditional organizations may have a more difficult time implementing agile (Lee et al. 2026), our study explains this through the lens of institutional pluralism and dynamism, which provide middle-range constructs for understanding institutional contexts as they apply to agile practices.

Our insights can also inform management research more broadly. Increasingly, organizations have adopted agile practices as general modes of operation (Berente 2020). As this study shows, agile practices are adopted differently across distinct institutional contexts. This is consistent with research on how institutional logics can shape general practices as varied as accounting (Quattrone 2015) and sustainability reporting (Mahmood and Uddin 2021). We offer dynamism and pluralism as two operational facets of institutional contexts that can predict how agile is translated locally, and ratcheting as a nuanced view of how agile change unfolds. Microlevel practice adaptations (ratcheting) serve as a mechanism linking macrolevel institutional pressures to the evolution of organizational routines. This middle-range theorizing helps bridge idiosyncratic practice variation to broader social regularities (Berente and Yoo 2012).

For practitioners, our study suggests that one can anticipate a drift in agile practices depending on the environment. Managers in pluralistic environments, for example, should not be surprised if documentation increases over time, and they may even plan for it in agile rollouts. Conversely, in fast-moving contexts, managers might embrace more flexibility and resist overstructuring the process.

Limitations

This research is not without limitations. In particular, the access we had to the research sites and the resulting number of interviews at each location could have been more extensive. We would have liked more data. However, it is important to remember that this is an exploratory study grounded in data with the intention to construct theory. Our inferences are well supported by the data and informed the creation of this perspective but could indeed benefit from elaboration. It is important to note that we are laying the groundwork for research on agile development as it is translated across contexts. We are building on and elaborating the work of, for example, Pries-Heje and Baskerville (2017), and future work can extend ours. As Online Appendix A shows, much of the existing work on tailoring is prescriptive, with little descriptive fieldwork on how agile is actually translated. Given the importance of agile in an increasing number of contexts, this is critical. Our research is a brick in the edifice of our knowledge (Tiwana and Kim 2019)—an exercise in constructing theory and not the final word. Future research should look to more contexts to elaborate on and extend this work.

In this research, we were not looking to explore how the social and power dynamics unfolded, but this would be a fascinating area for study. Further, recent work emphasizes the congruence of agile practices with individual developers (Benlian 2022, Benlian et al. 2025). Translation theory highlights how individual developers can be heterogeneous in their enactment of practices even in the same team (Waldorff and Madsen 2023), and different people serve different roles in a translation ecosystem (Nielsen et al. 2022, Elmholdt et al. 2025) Future research rooted in our middle-range perspective can help to link this ecological perspective with individual actions.

Further, we sampled cases based on their claims of implementing agile development, but there is a wide variety of approaches labeled agile. Just as we differentiate among types of contexts, agile development can mean many things, and descriptive accounts should be clear about this. Finally, our research emphasized four general dimensions of agile processes that we developed through our coding and constant comparison with the literature. Although we do not capture everything about their agile processes, we address what we saw changing. Thus, future research may unpack some of these practices or add others to better understand how context matters for translating agile.

Conclusion

Agile development is not a one-size-fits-all approach; its practices are translated in ways that are patterned by the institutional contexts in which they are embedded. Our study introduces a perspective for understanding these patterns, highlighting that the journey of agile adoption involves ongoing adjustments—a ratcheting process—that align with an organization's dynamism and pluralism. This nuanced understanding helps set realistic expectations and provides a foundation for future research to further unravel how context and method interact in agile development.

Endnotes

[1] Agile practices identified vary across studies; for example, Diebold and Dahlem (2014) identified 18 agile practices, and Sidky et al. (2007) identify 40 agile practices. Different studies list different sets of agile practices depending on the goals of the research.

[2] These dimensions were generated from empirical coding of changing practices and constant comparison with the literature. They are presented in the review of literature to allow scaffolding for the reader, not as a deductive, linear application of these categories.

[3] Middle-range theory is a term coined by sociologist Robert Merton to refer to theories that "are abstract enough to allow for generalizations, but close enough to observed data in order to be incorporated into propositions that can be empirically validated" (Hassan and Lowry 2015, p. 6). In this sense, we seek to generalize specifically about agile and constructs capturing institutional characteristics (dynamism, pluralism) while abstracting away from the wide variety of organizational contexts and agile practices.

References

Abrahamsson P, Babar MA, Kruchten P (2010) Agility and architecture: Can they coexist? *IEEE Software* 27(2):16–22.

Annosi MC, Mattarelli E, Micelotta E, Martini A (2022) Logics' shift and depletion of innovation: A multi-level study of agile use in a multinational telco company. *Inform. Organ.* 32(3):100421.

Anupindi R, Chopra S, Deshmukh SD, Van Mieghem JA, Zemel E (2006) *Managing Business Process Flows: Principles of Operations Management* (Pearson Prentice Hall, Upper Saddle River, NJ).

Avgerou C (2000) IT and organizational change: An institutionalist perspective. *Inform. Tech. People* 13(4):234–262.

Avison DE, Fitzgerald G (2003) Where now for development methodologies? *Comm. ACM* 46(1):78–82.

Avison DE, Wood-Harper AT (1991) Information systems development research: An exploration of ideas in practice. *Comput. J.* 34(2):98–112.

Baham C, Hirschheim R (2022) Issues, challenges, and a proposed theoretical core of agile software development research. *Inform. Systems J.* 32(1):103–129.

Barroca L, Dingsøyr T, Mikalsen M (2019) Agile transformation: A summary and research agenda from the first international workshop. Hoda R, ed. *Agile Processes in Software Engineering and Extreme Programming—Workshops*, Lecture Notes in Business Information Processing, vol. 364 (Springer, Cham, Switzerland), 3–9.

Bass JM (2016) Artefacts and agile method tailoring in large-scale offshore software development programmes. *Inform. Software Tech.* 75:1–16.

Baxter D, Dacre N, Dong H, Ceylan S (2023) Institutional challenges in agile adoption: Evidence from a public sector IT project. *Government Inform. Quart.* 40(4):101858.

Beck K, Andres C (2004) *Extreme Programming Explained: Embrace Change*, 2nd ed. (Addison-Wesley Professional, Boston).

Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, et al. (2001) The agile manifesto. Accessed September 20, 2008, http://www.agilemanifesto.org/.

Benlian A (2022) Sprint zeal or sprint fatigue? The benefits and burdens of agile ISD practices use for developer well-being. *Inform. Systems Res.* 33(2):557–578.

Benlian A, Pinski M, Adam M (2025) Team-enacted use vs. developer-needed use of agile practices: How perceptual (in-)congruence and team feedback-seeking shape developer well-being. *Inform. Systems Res.* 36(4):2253–2277.

Berente N (2020) Agile development as the root metaphor for strategy in digital innovation. Nambisan S, Lyytinen K, Yoo Y, eds. *Handbook of Digital Innovation* (Edward Elgar Publishing, Cheltenham, UK), 83–96.

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

Berente N, Lyytinen K, Yoo Y, Maurer C (2019) Institutional logics and pluralistic responses to enterprise system implementation: A qualitative meta-analysis. *MIS Quart.* 43(3):873–902.

Bianchi M, Marzi G, Dabić M (2022) Guest editorial: Agile beyond software—In search of flexibility in a wide range of innovation projects and industries. *IEEE Trans. Engrg. Management* 69(6):3454–3458.

Birks DF, Fernandez W, Levina N, Nasirin S (2013) Grounded theory method in information systems research: Its nature, diversity and opportunities. *Eur. J. Inform. Systems* 22(1):1–8.

Boland RJ, Lyytinen K, Yoo Y (2007) Wakes of innovation in project networks: The case of digital 3-D representations in architecture, engineering, and construction. *Organ. Sci.* 18(4):631–647.

Brinkkemper S (1996) Method engineering: Engineering of information systems development methods and tools. *Inform. Software Tech.* 38(4):275–280.

Campanelli AS, Camilo RD, Parreiras FS (2018) The impact of tailoring criteria on agile practices adoption: A survey with novice agile practitioners in Brazil. *J. Systems Software* 137:366–379.

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

Clark HH (1996) *Using Language* (Cambridge University Press, Cambridge, UK).

Cockburn A (2002) *Agile Software Development* (Addison-Wesley, Boston).

Cockburn A (2004) *Crystal Clear: A Human-Powered Methodology for Small Teams* (Addison-Wesley Professional, Upper Saddle River, NJ).

Coleman G, O'Connor R (2008) Investigating software process in practice: A grounded theory perspective. *J. Systems Software* 81(5):772–784.

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

Conboy K, Fitzgerald B (2010) Method and developer characteristics for effective agile method tailoring: A study of XP expert opinion. *ACM Trans. Software Engrg. Methodology* 20(1):2.

Corvera Charaf M, Rosenkranz C, Holten R (2013) The emergence of shared understanding: Applying functional pragmatics to study the requirements development process. *Inform. Systems J.* 23(2):115–135.

Cram WA, Newell S (2016) Mindful revolution or mindless trend? Examining agile development as a management fashion. *Eur. J. Inform. Systems* 25(2):154–169.

Czarniawska B, Mazza C (2003) Consulting as a liminal space. *Human Relations* 56(3):267–290.

Czarniawska B, Sevón G (1996) *Translating Organizational Change* (Walter de Gruyter & Co., Berlin).

Diebold P, Dahlem M (2014) Agile practices in practice: A mapping study. Shepperd MJ, Hall T, Myrtveit I, eds. *Proc. 18th Internat. Conf. Evaluation Assessment Software Engrg.*, vol. 30 (Association for Computing Machinery, New York), 1–10.

DiMaggio PJ, Powell WW (1983) The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *Amer. Sociol. Rev.* 48(2):147–160.

Dybå T, Dingsøyr T (2009) What do we know about agile software development? *IEEE Software* 26(5):6–9.

Elmholdt KT, Nielsen JA, Wæraas A, Meyer R (2025) It takes a village: Translating management ideas through an ecology of roles. *J. Management Stud.* 62(7):2938–2968.

Fitzgerald B, Hartnett G, Conboy K (2006) Customising agile methods to software practices at Intel Shannon. *Eur. J. Inform. Systems* 15(2):200–213.

Fowler M, Highsmith J (2001) The agile manifesto. *Software Development* 9(8):28–35.

Friedland R, Alford RR (1991) Bringing society back in: Symbols, practices, and institutional contradictions. Powell WW, DiMaggio PJ, eds. *The New Institutionalism in Organizational Analysis* (University of Chicago Press, Chicago), 232–263.

Giardino C, Paternoster N, Unterkalmsteiner M, Gorschek T, Abrahamsson P (2016) Software development in startup companies: The greenfield startup model. *IEEE Trans. Software Engrg.* 42(6):585–604.

Gioia DA, Corley KG, Hamilton AL (2013) Seeking qualitative rigor in inductive research: Notes on the Gioia methodology. *Organ. Res. Methods* 16(1):15–31.

Gregory RW, Beck R, Keil M (2013) Control balancing in information systems development offshoring projects. *MIS Quart.* 37(4):1211–1232.

Gustavsson T (2021) Institutional logics in large-scale agile software development Transformations. Gregory P, Kruchten P, eds. *Agile Processes in Software Engineering and Extreme Programming—Workshops* (Springer International Publishing, Cham, Switzerland), 12–19.

Hahn J, Zhou J, Lee G, Zorin V (2025) Organizing for software product development: The effects of team structure, product complexity, and cross-team coordination. *Inform. Systems Res.*, ePub ahead of print June 2, https://doi.org/10.1287/isre.2023.0154.

Hansen S, Baroody AJ (2020) Electronic health records and the logics of care: Complementarity and conflict in the U.S. healthcare system. *Inform. Systems Res.* 31(1):57–75.

Hassan N, Lowry P (2015) Seeking middle-range theories in information systems research. *ICIS 2015 Proc.*, https://aisel.aisnet.org/icis2015/proceedings/IStheory/1.

Henderson-Sellers B, Ralyté J, Ågerfalk PJ, Rossi M (2014) *Situational Method Engineering* (Springer, Berlin).

Hoda R, Noble J, Marshall S (2012) Developing a grounded theory to explain the practices of self-organizing Agile teams. *Empirical Software Engrg.* 17(6):609–639.

Hummel M, Rosenkranz C, Holten R (2013) The role of communication in agile systems development. *Bus. Inform. System Engrg.* 5:343–355.

Hummel M, Rosenkranz C, Holten R (2015) The role of social agile practices for direct and indirect communication in information systems development teams. *Comm. Assoc. Inform. Systems* 36: 273–300.

Iivari J (1989) A methodology for IS development as organizational change: A pragmatic contingency approach. Klein H, Kumar K, eds. *Systems Development for Human Progress* (North-Holland, Amsterdam), 197–217.

Iivari J, Iivari N (2011) The relationship between organizational culture and the deployment of agile methods. *Inform. Software Tech.* 53(5):509–520.

Kellogg KC (2022) Local adaptation without work intensification: Experimentalist governance of digital technology for mutually beneficial role reconfiguration in organizations. *Organ. Sci.* 33(2):571–599.

Kudaravalli S, Faraj S, Johnson SL (2017) A configural approach to coordinating expertise in software development teams. *MIS Quart.* 41(1):43–64.

Kum ME, Kortmann S, Perols JL, Zimmermann C (2024) How digitization and agile methods influence corporate venture

performance: An organizational information-processing perspective. *IEEE Trans. Engrg. Management* 71:11024–11038.

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

Larman C (2003) *Agile and Iterative Development: A Manager's Guide* (Addison-Wesley, Boston).

Lee G, Xia W (2010) Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quart.* 34(1):87–114.

Lee JS, Maruping LM, Keil M (2026) Understanding factors affecting the use of agile development practices in system of systems environments. *J. Assoc. Inform. Systems.* Forthcoming.

Lyytinen K, Newman M (2008) Explaining information systems change: A punctuated socio-technical change model. *Eur. J. Inform. Systems* 17(6):589–613.

Mahmood Z, Uddin S (2021) Institutional logics and practice variations in sustainability reporting: Evidence from an emerging field. *Accounting, Auditing Accountability J.* 34(5):1163–1189.

Maruping LM, Venkatesh V, Agarwal R (2009) A control theory perspective on agile methodology use and changing user requirements. *Inform. Systems Res.* 20(3):377–399.

Moe NB, Fægri TE, Cruzes DS, Faugstad JE (2016) Enabling knowledge sharing in agile virtual teams. *Proc. 2016 IEEE 11th Internat. Conf. Global Software Engrg.* (Institute of Electrical and Electronics Engineers, Piscataway, NJ).

Morris T, Lancaster Z (2006) Translating management ideas. *Organ. Stud.* 27(2):207–233.

Neumann O, Kirklies PC, Schott C (2024) Adopting agile in government: A comparative case study. *Public Management Rev.* 26(12):3692–3714.

Nielsen JA, Mathiassen L, Newell S (2014) Theorization and translation in information technology institutionalization: Evidence from Danish home care. *MIS Quart.* 38(1):165–186.

Nielsen JA, Mathiassen L, Newell S (2022) Multidirectional idea travelling across an organizational field. *Organ. Stud.* 43(6): 931–952.

Pantiuchina J, Mondini M, Khanna D, Wang X, Abrahamsson P (2017) Are software startups applying agile practices? The state of the practice from a large survey. Baumeister H, Lichter H, Riebisch M, eds. *Agile Processes in Software Engineering and Extreme Programming*, Lecture Notes in Business Information Processing, vol. 283 (Springer, Cham, Switzerland), 167–183.

Pikkarainen M, Haikara J, Salo O, Abrahamsson P, Still J (2008) The impact of agile practices on communication in software development. *Empirical Software Engrg.* 13(3):303–337.

Poppendieck M, Cusumano MA (2012) Lean software development: A tutorial. *IEEE Software* 29(5):26–32.

Pries-Heje J, Baskerville R (2017) The translation and adaptation of agile methods: A discourse of fragmentation and articulation. *Inform. Tech. People* 30(2):396–423.

Qiu Y, Gopal A, Hann I-H (2017) Logic pluralism in mobile platform ecosystems: A study of indie app developers on the iOS app store. *Inform. Systems Res.* 28(2):225–249.

Quattrone P (2015) Governing social orders, unfolding rationality, and Jesuit accounting practices: A procedural approach to institutional logics. *Admin. Sci. Quart.* 60(3):411–445.

Ramesh B, Cao L, Kim J, Mohan K, James TL (2019) Consider culture when implementing agile practices. *MIT Sloan Management Rev.* 60(3):1–37.

Recker J, Holten R, Hummel M, Rosenkranz C (2017) How agile practices impact customer responsiveness and development success: A field study. *Project Management J.* 48(2):99–121.

Reineke P, Katila R, Eisenhardt KM (2025) Decentralization in organizations: A revolution or a mirage? *Acad. Management Ann.* 19(1):298–342.

Ries E (2011) *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses* (Crown Books, New York).

Rolland KH, Mathiassen L, Rai A (2018) Managing digital platforms in user organizations: The interactions between digital options and digital debt. *Inform. Systems Res.* 29(2):419–443.

Sarasvathy SD (2001) Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. *Acad. Management Rev.* 26(2):243–263.

Sarker S, Sarker S (2009) Exploring agility in distributed information systems development teams: An interpretive study in an offshoring context. *Inform. Systems Res.* 20(3):440–461.

Schwaber K, Beedle M (2002) *Agile Software Development with Scrum* (Prentice Hall, Upper Saddle River, NJ).

Sidky A, Arthur J, Bohner S (2007) A disciplined approach to adopting agile practices: The agile adoption framework. *Innovation Systems Software Engrg.* 3:203–216.

Strauss A, Corbin J (1998) *Basics of Qualitative Research: Procedures and Techniques for Developing Grounded Theory* (SAGE, Thousand Oaks, CA).

Thornton PH, Ocasio W (2008) Institutional logics. Greenwood R, Oliver C, Suddaby R, eds. *The Sage Handbook of Organizational Institutionalism* (SAGE Publications, Thousand Oaks, CA), 99–129.

Thornton PH, Ocasio W, Lounsbury M (2012) *The Institutional Logics Perspective: A New Approach to Culture, Structure, and Process* (Oxford University Press, Oxford, UK).

Tiwana A, Kim SK (2019) From bricks to an edifice: Cultivating strong inference in information systems research. *Inform. Systems Res.* 30(3):1029–1036.

Tiwana A, Konsynski B (2010) Complementarities between organizational IT architecture and governance structure. *Inform. Systems Res.* 21(2):288–304.

Tolfo C, Wazlawick RS, Ferreira MGG, Forcellini FA (2011) Agile methods and organizational culture: Reflections about cultural levels. *J. Software Maintenance Evolution: Res. Practice* 23(6):423–441.

Tyre MJ, Orlikowski WJ (1994) Windows of opportunity: Temporal patterns of technological adaptation in organizations. *Organ. Sci.* 5(1):98–118.

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.

Wæraas A, Nielsen JA (2016) Translation theory "translated": Three perspectives on translation in organizational research. *Internat. J. Management Rev.* 18(3):236–270.

Waldorff SB, Madsen MH (2023) Translating to maintain existing practices: Micro-tactics in the implementation of a new management concept. *Organ. Stud.* 44(3):427–450.

Wang X, Conboy K, Pikkarainen M (2012) Assimilation of agile practices in use. *Inform. Systems J.* 22(6):435–455.

Yoo Y, Henfridsson O, Lyytinen K (2010) Research commentary—The new organizing logic of digital innovation: An agenda for information systems research. *Inform. Systems Res.* 21(4):724–735.