Rule Formation and Change in Information Systems Development: How Institutional Logics Shape ISD Practices and Processes

Nicholas Berente
University of Georgia
berente@uga.edu

Sean W. Hansen
Rochester Institute of Technology
shansen@saunders.rit.edu

Christoph Rosenkranz
University of Cologne
rosenkranz@wiso.uni-koeln.de

Abstract
Information systems development methodologies are appropriated differently by different organizations. Drawing on an institutional logics perspective, we offer one explanation for rule differences across contexts. Variations in the local appropriation of ISD methods, and the ways rules are formed and changed, reflect the “institutional logics” of the particular context. In this study of three roughly equivalent agile ISD projects in different contexts (startup, consulting, and infrastructure), we identify three different institutional logics, and show how these logics shape the rules that are formed and changed in ISD projects. These findings emphasize the need to attend to institutional contexts of ISD projects when looking to understand the local enactment of rule formation and change.

1. Introduction
Information systems development (ISD) is an inherently complex sociotechnical process. ISD participants manage this complexity by adapting well-established approaches and methodologies to their local situations [1]. ISD methodologies are systematic approaches to the development of systems, including guidelines concerning tools, techniques, documentation, and the sequencing of tasks [2]. Methodologies embody the mechanisms for work, coordination, and control in an ISD project, as well as prescriptions for appropriate action – in short, what we refer to as “rules.” Rules are a central feature of coordination in complex social environments [3].

Rules are the norms, laws, regulations, taboos, customs, and related concepts that organize and regulate most human social activity. Rules are not static and are frequently adapted, changed, or even ignored. For example, ISD methodologies-in-use are often not enacted “by the book,” but can rather be characterized as a combination of rules and practices from different methods [4]. Although ISD teams often claim to employ a particular methodology, the specific rules adapted and appropriated can vary across contexts [5]. For example, research on ISD projects using the extreme programming (XP) approach suggests that few teams actually implement all of the rules of XP [6]. Many of the rules are adapted or changed over time to be operationalized in different situations [7] and to scale up or down to more or less complex projects [8, 9]. Although XP is popular and widely appropriated, its actual implementation can vary widely from organization to organization and even project to project.

Rules for coordinating ISD – both formal and informal – are adopted (and adapted) differently in different situations, and they evolve in idiosyncratic ways over time. Yet there is little understanding of how and why organizations formulate, adopt, and change the rules they use to coordinate ISD. One (largely prescriptive) explanation for this is that certain practices “fit” better with different contexts [10, 11]. We refer to this as the “contingency” perspective. The contingency perspective of ISD method adoption indicates that organizations should adopt the rules that address the complexity, size, and other factors associated with an ISD project. However, while this view does offer prescriptive recommendations, it does little to help us understand what actually happens in practice. ISD approaches can vary dramatically across very similar contexts and within a particular context over time.

In this paper we do not take a prescriptive view of ISD methods and rules, like many contingency arguments, but instead we look to understand how organizations formulate, adopt, and change the rules they use to coordinate ISD. To do so, we adopt an “institutional logics” perspective [13]. Institutional logics are standards for action rooted in broader social phenomena, and can be used to identify regularities across local practices [12]. We draw upon this perspective [13] to explain how the social context of ISD shapes the rules that constitute ISD practice. To do this, we explore three ISD projects undertaken in three fundamentally different
institutional contexts. We analyze how the evolution of rules in these projects reflects the logics of the different teams’ institutional contexts.

The remainder of the paper is organized as follows. First, we review the literature on ISD methodologies and rules, and introduce and apply the institutional logics perspective to rule evolution in ISD. We then present our study and conclude with an argument for investigating the role of institutional logics for rule formation and evolution in ISD.

2. ISD Rule Formation and Change

2.1. ISD Methods and Rules

The history of ISD can be characterized as the movement of a pendulum along an axis of process rule formality. The early days of ISD were marked by the absence of clear standards guiding the practice of coding and project documentation [14, 15] – the era of ad hoc development or “cowboy coding”. This was followed in the 1970s by the rise of structured development methodologies, in which phases of the ISD process are clearly demarcated and stringent standards for appropriate practice are established. Royce’s [16] model of software development, partitioning ISD into seven subsequent phases and later dubbed the “waterfall model,” is emblematic of this methodological approach. Rules associated with the waterfall model included different practices for each of the seven different phases. For example, one rule was that the design phase should come first, another was that program designers should initiate the process; and that there must be an “overview document” along with a variety of “data processing” plans developed in the program design phase.

Critics of this family of approaches argued that they did not allow for sufficient iteration and established (or reinforced) a separation between developers and system users [17]. This led to the development of more iterative models, such as Boehm’s [15] spiral model, prototyping [18], or rapid application development [19]. The spiral model, for example, included rules for multiple cycles of the ISD process. Each cycle involves prototyping, risk analysis and resolution, structured reviews, planning for the next phase, and so forth.

Concurrent with the development of the spiral model, there were humanistic, user-centered methodologies forming in the U.S. [20], U.K. [21], and Scandinavia [22]. These methods took an explicitly sociotechnical stance and looked to optimize the system under development by involving end users in the design of the system. This movement laid the groundwork for “participatory” approaches to ISD [23]. One such participatory approach, ETHICS, included a highly structured approach with a number of rules associated with user involvement and bottom-up participation, including definition of job satisfaction needs, forecasting future needs, identifying alternative solutions before coding, ranking of solutions, and so forth [21].

Dissatisfaction with the structured ISD approaches led to the publication of the Manifesto for Agile Software Development in 2001 [24]. Crafted by self-appointed software “anarchists,” the Manifesto articulates the group’s shared principles for overturning the prevailing orthodoxy of plan-driven ISD. Key rules of the resulting agile approaches to ISD included guidelines associated with close collaboration amongst design stakeholders, intensive iteration and rapid prototyping, openness to changing requirements, and the eschewal of “heavyweight” processes and formal documentation. Thus, the agile development label functions as an umbrella terms for several distinct methodologies organized around some common principles. Prominent agile methodologies include Scrum [25], XP [26], and Crystal [27]. Industry has adopted many agile methods in recent years.

As this brief overview illustrates, the substantive changes in the practice of ISD have focused on the question of rules governing software development. Each approach or “school of thought” advocates for a distinct set of rules, promoting structured vs. agile, participatory vs. top-down, plan-driven vs. self-organizing, or “heavyweight” vs. “lightweight.” Among all of the methodologies, a variety of techniques, capabilities, practices, and rules are available to ISD teams to conduct their projects [1].

2.2. Rule Formation and Change in ISD

ISD methodologies are generally chosen at the outset of a project. Methodologies are typically only partially adapted and can depart dramatically from their textbook characterizations. Organizations select practices and rules that they can employ as part of a broader approach to ISD. This often involves tailoring methodologies to a project setting [28]. Inevitably, this leads to the emergence, adoption, and use of multiple variations of a methodology. There is little understanding, however, about how and why organizations choose to adapt and tailor the methodologies that they use.

For us, an ISD methodology is essentially a set of rules – “shared understandings by actors about enforced prescriptions concerning what actions (or outcomes) are required, prohibited, or permitted” [29: 319]. Rules as enacted practices thereby includes the execution of prescriptive, formalized processes (e.g., test-driven development [30]) as well as the
adoption of emergent, informal routines (e.g., regular reflective discussions during meetings [31]). Since we know little about why and how different rules are formed, adapted, and changed in practice, the question is: how and why do organizations form, adopt, and change the rules for coordinating ISD?

Much of the prevailing literature reflects a prescriptive “contingency” perspective on the design and adoption of ISD methodologies. Such contingency approaches suggest that one should first consider the contextual characteristics of the environment, and then consider appropriate mechanisms out of a set of alternatives for dealing with the situational context at hand [32]. Contingency models in ISD [10, 11] offer categories of contextual factors and portfolios of practices, techniques, and approaches to ‘fit’ methodologies to contexts [33]. Several such contingency models have been created in the past, for example, to help develop software for decision support [34] or to reduce the uncertainty of requirements for software applications [35]. The challenge is to identify the relevant contextual factors and to create an overall consistency (or ‘fit’) between these contextual factors and the set of available development practices. For example, the selection and adoption of different ISD methodologies is typically perceived to depend on the complexity of the system under development and the characteristics and skills of the available team members.

However, there are two issues with the contingency view on ISD methodologies. First, the contingency perspective assumes a functionalist approach to ISD – development teams operating in isolation and making decision about methodology based on an “objective” assessment of contextual factors. We argue that in actuality, development groups often operate within a broader organizational context, and the institutional environment frames their decisions. For example, the ETHICS method [21] and the Scandinavian tradition in ISD are as much a reflection of a distinct set of values (i.e., emancipation and labor participation) as the material context of the ISD effort. Second, the contingency perspective assumes that rules remain relatively stable over the course of an ISD effort. Indeed, the evidence from practice suggests that rules change considerably throughout a project; each iteration or stage may involve the adoption of new practices, tools, ways of working, training, finance, or team structure [36]. The contingency approach thus offers limited value for explaining why rules form, change, and evolve in practice. Many factors such as team composition and task complexity remain relatively stable throughout a project – this is not the case for practices, methodologies-in-use, and associated rules.

It cannot be denied that ISD is a social process [37]. The social context of ISD most certainly has some impact on the practice of ISD, and an explicitly social-technical view of rule formation and change can aid our understanding of why rules change in ISD. An institutional view of IS phenomena can help tie local, idiosyncratic practice to broader societal institutions [12, 38]. Thus an expressly institutional view may provide leverage for exploring how rules are formed and changed in ISD practice.

2.3. Institutional Logics
Contemporary organizational institutionalism, or new institutionalism [39], provides a theoretical lens that allows us to explain the relationships between situated practices and organizational, cultural, and societal contexts [12, 38]. At a basic level, institutions are defined as organized, established procedures that are continually reinforced through their reenactment, persist over time, and are thus objectified in discourse [40]. Institutions provide the rules for action in the form of norms, scripts, prescriptions, and typifications associated with appropriate action in different situations. Individuals draw upon preexisting institutions to guide their action, and in enacting those institutional rules, they reinforce them over time. Thus, institutional rules evolve through changing local enactments [13]. Such change may be gradual, resulting from novel adaptation of rules to local situations [41], or dramatic, with intentional innovation [42].

Whether gradual or discontinuous, local enactments reflect the rationality of individuals pursuing the most appropriate actions given their identities and the goals, values, and assumptions that they bring to a given situation. As such, there is no single standard that is always appropriate in every situation. Instead, actors assess situations and look to different institutions to guide their actions. This recognition that human action is institutionally conditioned has been described as “embedded rationality” [43] – humans act rationally, but do so in accordance with the logic of reference institutions. An institutional logic is a set of goals, values, and prescriptions [13] which constitute the organizing principles of practices for a specific institution.

Action is always embedded in an institutional context [13]. Actors are faced with a situation and they draw upon the appropriate logics to identify the rules relevant to coordinating activity in that context. Drawing upon particular institutional logics can involve novelty and intentionality, translation, or direct enactment. In an ISD context, one would then expect some teams to draw upon different logics – even given similar, size, scope, team composition,
and complexity (i.e., contingency factors) – based on the institutional logics within which their practices are embedded. Essentially, we propose that different teams will use different logics in appropriating and changing rules across an ISD project.

3. Case Studies

3.1. Research Method
We used a multiple case study approach [44] to explore how rules change and evolve in different ISD settings. The three case organizations are described in more detail below. The two main methods for data collection were interviews and observation. We visited each organization multiple times for one to three days at each visit. We conducted interviews with selected informants, group interviews, and observations of activities. We used a structured interview protocol intended to uncover rules as well as events leading to rule changes. All interviews, which lasted from 60 to 120 minutes, and meetings were audio-recorded as well as transcribed.

We followed a two-stage process of inductive and deductive coding of data, building upon and adapting the recommendations by Miles and Huberman [45]. First, all three authors scrutinized and coded the data independently of each other. Inspired by previous work on rules and rule evolution [29], we started with initial seed codes and searched for evidence of institutional logics, rules, and rule evolution. Subsequently, all three authors discussed their interpretations in direct conversation. This resulted in the following analysis.

3.2. Case Overviews
Comparisons among multiple case sites demonstrate the influence of contextual variability and therefore yield broader research results than single cases [44]. Based on a literal replication strategy, we expect the conditions of the cases to lead to different results in for organizations in three different environments.

**Startup** is a U.S. social media aggregator that develops software for both internal and external clients. Their software is focused on the exploratory analysis of large amounts of web-based content (i.e., “big data”). **Startup** developers employ an agile development approach that is a variation on the Scrum agile methodology. The ISD teams operate on a three-week sprint cycle, conduct stand-up meetings two or three times weekly (rather than daily), and do not use a formal task board. The firm relies upon Amazon’s cloud services for all data storage and maintains their software in a GitHub repository.

The German company **Consult** employs about 60 employees in its software development department, of which 40 employees regularly use a mix of Scrum and XP practices in their work. Consulting in the financial sector generates the main income of the company. In order to take advantage of the knowledge that is acquired in the consulting activities, the company started to develop their own specialized software products for customers. Three years ago, the managers decided to start developing in an agile way in order to increase customer satisfaction and project performance. By the end of 2013, the software developers are not employing a specific agile ISD methodology such as Scrum or XP in a strict way. Instead, several agile practices of different agile ISD methodologies that fitted the organizational needs (e.g., daily stand-up meetings, continuous integration) have been adopted and combined with more formal, plan-driven practices (e.g., modeling requirements by use case diagrams).

**Infra** is a U.S. federally-funded digital infrastructure project. The organization develops elements of the digital platform that enables access to scientific resources, and associated services. The scheduling and resource allocation system addressed in this research has a more than ten year history. For the first decade, the development process was highly centralized – one manager was firmly in control and development was predominantly focused in a single location. With recent changes in organizational leadership (that manager is leaving and new top management has taken over), there has been increased formalization of the development process throughout the organization, and this has also affected this process (architecture-driven and use-case oriented). This formalization now allows for an object-oriented product architecture, that enables development to be distributed across the U.S. (concentrated in two main locations).

4. Analysis
We identified different institutional logics that are at play in each of our case organizations.

4.1. Startup
As the pseudonym suggests, **Startup** is the youngest of the organizations analyzed in our study. The firm operates in an emerging marketplace (social media aggregation and big data analytics), and the members have a strong focus on innovation and adaptation to changing market needs. We characterize the predominant institutional logic of **Startup** to be one of entrepreneurship, in which ad hoc organizing, bricolage, and flexibility are guiding principles.
With respect to ISD within Startup, the adoption of agile methods was an early (and self-reportedly obvious) decision given the firm’s adaptive ethos:

“We very rarely actually plan out two, three quarters in advance. What we know is the yearly strategic arc and we know what we are going to do in Q1. So what do we do in Q1 to meet that arc, right? So we conceptualize it, and then we replan every quarter to see what we achieved and what adjustments need to be made. So it’s very dynamic and very agile in the sense we review every quarter.” – CTO

In particular, the firm decided to adopt the Scrum methodology. To some degree, this selection was based on the past experiences of the firm’s CTO:

“We use Scrum. I actually worked with the initiator of agile at [an ecommerce firm]. He defined what Scrum was and how agile should be done, and my group was one of the first in [the firm] to adopt that framework.” – CTO

Startup development teams initially tried to adhere to practices advocated in the Scrum methodology, including brief daily stand-up meetings (“scrum”), proscribed sprint durations, development teams of eight or fewer. However, this rule set was changed in several key regards based on the experiences of the developers. Key changes included the elimination of daily scrums, the movement from two-week to three-week sprints, and the adoption of even-tighter constraints on team size:

**Daily Scrums:** “We do not do daily stand-ups, which is part of the Scrum process, because the reason is when you have two people working on it together, they know what’s going on and you know it’s really an old hat to have like you know 25 people stand around talking about what they did every day.” – Engineer 1

**Sprint Duration:** “We don’t do two-week sprints. I think that’s the general practice for many other companies, but doing sprint planning every week or every two weeks was a bit too much for my guys and they said ‘No. We want it to be three weeks. It gives us enough time to finish something and release it.’” – CTO

**Team Size:** “[We] break the work down into smaller chunks that can be achieved by no more than two people. We used to call this a one-pizza or a two-pizza team – the idea is you can feed them with one or two boxes of pizza. At Startup, it’s almost always one-pizza teams, but it’s just two people, plus a Product Manager.” – Product Manager

The changes to the methodological rule set at Startup are reflective of the firm’s emphasis on adaptation and doing “whatever works.” It is also indicative of a decentralized decision-making structure. The firm prides itself on hiring independent thinkers and empowering them to initiate action:

“I like to give that freedom to the team because when you hire the kind of people we do, then you don’t want to be you know super pedantic about all of this. You let the team make the decision and as long as everyone understands the order or goals and why they’re doing all this, then we are in good shape.” – CTO

4.2. Consult

For Consult, our data shows that ISD is strongly influenced by an institutional logic of a consulting profession, which is customer-centric and emphasizes high product quality from the customer perspective:

“We are not in the position that we have to deliver our software very quickly to the customer. It is more necessary that the software really fits to one special customer, and the software has to be very flexible and customizable, and we do not have a wide range of customers, but a small range of customers with special needs, which must be covered as well as possible.” – Product Manager B

Project outcomes are therefore also judged by measuring quality in terms of customer satisfaction, and Consult’s ISD process involves iterating with the customer until the customer is satisfied:

“We are playing rounds and rounds with the experts to really make nice design of the software that are implemented afterwards.” – Product Manager A

“[Quality is measured] especially by customer feedback. So we have a couple of customers. We have internal customers and external customers and moreover we are measuring the quality of the amount of bugs that are reported after the release. [...] but I think 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.’ And try to get more into the direction of the customer’s expectations and so we always try to go very early to the customer [to] show the new features and try to put the feedback into the software.” – Product Manager B

On the one hand, this leads to an emphasis on control and monitoring within projects to balance the scope
of the project in terms of effort and resources with the needs of the customer:

“The bigger the project is, the harder it is to keep those teams free from any steering … With our big project, there were ten people working on this. So that means 5 days, 10 people, 50 days. So in our internal calculation this means roughly about 50 thousand Euros of costs. When they work in the wrong direction, the management wants to keep track of this. So having a team that is running three or four weeks without any chance of steering was really hard for me and my colleagues, seeing after one month [the result] wasn’t what we expected.” – Product Manager A

This importance of balancing and of control is reflected in tight management of project budgets and documenting requirements. For example, upfront milestone planning is important for steering and managerial control:

“There was no real detailed plan of what [developers] should deliver at which point in time, but there was a rough milestone-plan, with milestones for two or three months and rough estimation upfront. But what we saw was always that the time schedule for those rough milestones was always shifted. … What we did years before was really a strong tracking … there was huge overhead of tracking and controlling. And we wanted to get rid of this. To really make the people feel that they can work in their own responsibility… because what we expected was more responsibility of the single person involved in the project team.” – Product Manager A

On the other hand, we also find that the customer is central to logics used in rule formation. For example, an important rule posits that new features in the product backlog are always created based on customer feedback as a key source, and subsequently are prioritized with regard to customer needs:

“Normally we have a couple of sources for the backlog. One source is the customer or the customizer, who works together with the customer. And they are creating change requests or defaulting bugs and create features. […] But normally [me and the project manager] are coming together for one or two hours at the beginning of an iteration and discuss about the tickets. What is necessary? What is the priority of the tickets?” – Product Manager B

This also leads to another rule adoption with regard to the duration of the iterations, or sprints, within the development. The usual sprint duration from Scrum has been adopted from fixed two to loose two- to four-week iteration cycles due to perceived quality problems:

“[The discussion of features] takes some time at the beginning of the iteration as well. So this is another reason why we have decided to go on four weeks instead of two weeks because you have two or three hours preparing and discussing the prioritization of features and the iterations.” – Product Manager B

“And this where we really work in sprints of two or four weeks where features are defined upfront and the project team is doing those features then within this time. … And for the next sprint maybe you can change it. Most [of the] time it’s like when you do the requirements part more in a project then you need smaller sprints or iterations like we call it. But it’s really like a sprint. So when you only do analysis then you want to see, ok in which direction does proceed and then we do the stone two weeks. But if we have bigger features to implement for incidence there were somehow features that had large amounts of work to be done to really see something going. And we not usually say ok, let’s be about one month. So having those close to four weeks.” – Product Manager A

4.3. Infra

For the first decade of development of the scheduling system, the process was ad hoc and there was little documentation – what existed was informal. The process was marked by periodic minor releases when new functionality was complete:

“… it’s never been one of those rigid, you have to test this before you can check it into CDS. We would do things, we would test it, and then we would put it out there and say hey, here it is.” – Developer A

A single manager controlled the project and determined what would be in each release:

“[Manager] did most of the prioritization. There was a balancing act between what could we do very quickly and get out to the users very quickly and then what was a longer term like mini project within the project … [requests] would come mostly through [Manager] and then we had a wish list that we would just chug away at for quite a while.” – Developer A

This centralized structure and informal development process led to a monolithic and tightly integrated architecture that was well-understood by the development team who had been working on it for over a decade, but was not documented nor easily extensible to fit with other projects in the enterprise.
This led to upper management making a decision to change how the scheduling system was organized:

“\[I\]n[Finally think the upper management at Infra recognized the growing importance of dealing with [scheduler] as a key piece of the Infra infrastructure that needed some attention...I mean I think probably out of the quarterly meeting there was somebody of a high enough position and it was probably a combination of [Infra Principle Investigator, or ‘PI’] and [Infra Co-PI], you know, the two co-PIs, who said this shall be done... Sort it out, yeah.” – Developer C

Development for the rest of the organization was distributed and took place primarily in Location2, which worked on the user interface of the digital infrastructure platform, whereas Scheduler was originally developed in Location1. It was important to bring the user interface in line with the rest of the infrastructure, but this was difficult do to the monolithic, undocumented code:

“I think that kind of tried to stress the importance of having an integrated interface, having one place where users go to, being able to automatically profile information, things like that. To have the entire process streamlined and easier for users through the user portal.” – Manager B

The result was adoption of a formal, documentation-driven process that the rest of the organization had begun using:

“The architecture folks have a fairly high level process and we’re trying to follow the major steps and then within each of those steps adapting their process. So you know, their process started with use cases and a high level architecture, they provided templates ... well actually, it started with the activity plan. And then they said use cases and a design document and we’ve been following that high level structure and then taking what they offer in terms of templates for the activity plan, the use cases, and adapting them to what we feel fits our current project and our current step. So the design document is the next major deliverable...” – Developer C

By formalizing the process, requiring strict adherence to the new architecture, and requiring a great deal of documentation, Infra was able to distribute the project – Location2 can now work on the user interface, while Location1 updates the back-end:

“The current effort will allow Location2 to do a lot more work independently because of the separation between the GUI and the back end with this API so, it’s very important for Location2 to move, make progress, and have something to work against... the use cases were pretty well divided into, you know, this is a system level integration, this is a user interface element, this is a review interface element, this is an admin interface element. So they’re pretty well divided up into, you know, the way these things work. And then we’re able to say this is part of back end, this is part of the admin interface, this is part of the review interface, this is part of the submit interface, this is part of the API interface, and stuff like that.” – Developer D

The result was enabling distribution and extensiveness:

“We want a packaged, solid thing. We want it to be a platform that we can build on top of and extend. And we want it to be well-defined interfaces, good documentation because you know like that’s another thing with ... that meeting that they had was just to go over the documentation because they couldn’t point us to a web page for the documentation because it didn’t exist.” – Developer C

5. Discussion

It has long been understood that ISD projects are social as much as they are technical [20]. Thus, the varying ISD methods introduced over the years attend to different facets of social interaction [23], resulting in dramatic changes to the nature of ISD within organizations [2]. Although there is a good deal of research on social processes associated with ISD (both normative and descriptive), there has been little emphasis on connecting local practices to the broader historical, political, and institutional contexts of ISD projects [46].

In this paper we take a first step toward connecting ISD methods to the broader institutional context using the concept of institutional logics. By conceiving ISD rules as institutional scripts for practice, we highlight how these scripts can be more or less consistent with different fields of practice. As such, we find that there is no ‘one correct way’ of adopting different rules for ISD. No one set of universal contingency factors (size, complexity, etc.) alone determines how ISD methods will be appropriated in practice. The organizations analyzed are fairly equivalent in size, complexity, overall espoused method (i.e., agile), and so forth. Differences in rule formation and change could not be accounted for by these factors. Rather, we find that the rules that were formed around practices in situ reflect different standards depending upon the
institutional logics the teams draw upon. We identify three logics in the cases, and we briefly address each.

Entrepreneurial Logic: Startup drew on a logic of leanness, informality, bricolage, and adaptability associated with entrepreneurial scripts for practice in determining when and how sprints will occur. Entrepreneurship is a well-established institution that is core to market practices in many countries—particularly the U.S. [47]. The scripts associated with entrepreneurship in the U.S. are tied to a bootstrapping, lean, informal focus on what needs to get done in order to grow and gain legitimacy. At any point an entrepreneur needs to be ready to respond and “pivot” in a new direction and therefore cannot realistically plan too far in advance. Startup’s rule changes reflected the informality and responsive thinking of the entrepreneur.

Consulting Profession Logic: Consult focused on high quality outcomes for specific customers, referencing the customer-centric logic of the consulting profession to structure iterations. Professions are central institutional actors in contemporary fields [48], and the consulting profession has a well-established customer-centric outlook. The goal of any consulting engagement involves addressing the particular needs of the client to render value. As such, quality (and subsequent engagements) is emphasized, and constant monitoring of progress from the customer perspective (e.g., budget) is essential to rule evolution in the adaptation of ISD methods. Although all three cases have important stakeholders, it was in the Consult case that the customer was front-and-center.

Managerialist Logic: Infra ISD rule changes reflected the managerial control and formality associated with bureaucratic rationalism. When the project was located in one place, it was closely controlled by a single manager. Senior leaders chose to distribute the project across locations, and coordinated their managerial authority through formalized practices and defined system architectures—an approach consistent with the managerial rationalism that dominated the organization [12]. The method appropriated after the change was agile and highly iterative, but it was shaped by a level of formality and rigidity (i.e., via structured canonical use cases, approval processes, defined architecture, etc.) not seen in projects in the other two case sites.

Together, the three cases imply a number of continua for how rules are formed and changed in local contexts. For example, Startup was the least formal, Infra was the most formal, and Consult was positioned somewhere in the middle. All three adopted agile practices and all three involved important (i.e., “mission critical”) applications, but the institutional context shaped the degree to which this formalization was reflected in the actual ISD process. Startup needed very little, Consult utilized formalization to the extent that it garnered customer feedback and controlled project progress, and Infra was highly formal. Similar continua may be developed for the pace or temporal rhythms of iterations, degree of developer discretion, etc.

Several studies have found the same technology implemented in highly similar organizational settings to be associated with very different consequences for structure and process (e.g., [49]). Transferred to ISD, the same ISD practices implemented in different organizational settings may be associated with different consequences for the ISD process. In this research we offer one explanation for why this occurs, and why different appropriations can be observed. The contexts of different ISD projects can involve different fields, professions, and institutionalized practices. The notion of an “institutional logic” captures the way broader scripts for appropriate action find their way into local practice. Through an analysis of the logics of different ISD efforts, we highlight how idiosyncratic local practice can reflect broader regularities in institutional fields.

In our investigation, we were led by the question of how and why different rules emerge in ISD. The application of an institutional logics perspective within multiple ISD efforts gives evidence for the practicability and utility of our concepts.

This lens offers several advantages to researchers and practitioners interested in the social dynamics of ISD. First, we offer a mechanism for carrying out explanation and prediction. The three cases illustrate that institutional logics with a focus on rule formation can serve as a lens for analyzing activities in ISD processes. Second, the different components and rules can be compared within and between settings and cases. Third, an institutional logics and rules perspective is capable of guiding both research and practice of ISD. If a sound understanding of ISD as a social process is desired, researchers and practitioners may benefit from our insights for conceptualizing the emergence of rules in a way that accounts for the institutional context of different projects. This lens avoids universalist assumptions of many contingency approaches to ISD method appropriation.

6. Conclusion

There is a general dearth of ISD research that goes beyond constructing or investigating specific methodologies, and instead looks to how ISD methodologies are actually appropriated in practice.
As Kautz et al. [46] note, “theory and studies of (longitudinal processes of) organization, specialization and institutionalization in ISD are needed” (p. 235). In this paper we propose an institutional logics perspective on ISD rule formation and change. The goal of this perspective is to help scholars, managers, and developers to understand regularities in the idiosyncratic enactment of methodologies through rule formation and change in ISD projects. These regularities reflect the institutional contexts within which ISD teams are embedded.

7. References


