

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection Lee Kong Chian School Of
Business

Lee Kong Chian School of Business

12-2019

From actions to paths to patterning: Toward a dynamic theory of patterning in routines

Kenneth T. GOH

Singapore Management University, kennethgoh@smu.edu.sg

Brian T. PENTLAND

Michigan State University

Follow this and additional works at: https://ink.library.smu.edu.sg/lkcsb_research



Part of the [Management Sciences and Quantitative Methods Commons](#), and the [Strategic Management Policy Commons](#)

Citation

GOH, Kenneth T. and PENTLAND, Brian T.. From actions to paths to patterning: Toward a dynamic theory of patterning in routines. (2019). *Academy of Management Journal*. 62, (6), 1901-1929.

Available at: https://ink.library.smu.edu.sg/lkcsb_research/6400

This Journal Article is brought to you for free and open access by the Lee Kong Chian School of Business at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection Lee Kong Chian School Of Business by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

FROM ACTIONS TO PATHS TO PATTERNING: TOWARD A DYNAMIC THEORY OF PATTERNING IN ROUTINES

KENNETH T. GOH
Singapore Management University

BRIAN T. PENTLAND
Michigan State University

This paper demonstrates a new way of seeing and theorizing about the dynamics of organizational routines through the concept of paths—time-ordered sequences of actions or events in performing work. Empirically and conceptually, paths provide the missing link between specific actions and patterns of action. When routines are represented as a narrative network, tracing the formation and dissolution of action paths can generate new insights about the dynamic patterning of actions in routine performances. We traced action paths using longitudinal field data from a videogame development project, and found that action patterns change dramatically over time based on the needs of the project. We explain these changes in terms of generic mechanisms that lead to the enactment of more (or fewer) paths in the narrative network. We propose that patterning can be seen as a new motor of routine dynamics and discuss generic mechanisms through which patterning can influence narrative network structure.

When we look at an organization, it is easy to see the people, places, departments, and other material and symbolic manifestations. Conceptually, however, we know that organizations are constituted by the continual unfolding and patterning of actions and interactions between these parts over time (Feldman, 2016a; Tsoukas & Chia, 2002; Weick, 1979). There is a gap between a processual view, which emphasizes patterns of action, and conventional ways of seeing and talking about organizations as collections of objects (Mesle & Dibben, 2016). Current theory tells us

We gratefully acknowledge the thoughtful feedback from Deputy Editor Pratima Bansal, three exceptional reviewers, and participants in the 2017 *Academy of Management Journal* “New Ways of Seeing” paper development workshop at the Ivey Business School, the 2018 Asian Management Research Consortium, the 2018 Academy of Management Big Data and Managing in a Digital Economy Special Conference, the 2018 Interdisciplinary Network for Group Research, and the Singapore Management University Strategy and Organisation group brown bag seminar series. We thank Anna Sycheva, Hashmat Habibzadah, Daniela Chang, and Gabriela Dedelli for their invaluable research assistance and acknowledge with deep gratitude the time our informants at GameSG dedicated to this project. We thank Carnegie Mellon University, Ivey Business School, Singapore Management University, and Michigan State University for financial support.

that processual phenomena are everywhere (Hernes, 2014; Langley & Tsoukas, 2016a), but they are harder to see (Feldman, 2016b).

In this paper, we introduce the concept of paths as a way of seeing and theorizing about the dynamics of organizational routines (Feldman, Pentland, D’Adderio, & Lazaric, 2016). By *path*, we mean a coherent, time-ordered sequence of actions or interactions in the workflow—steps in a process of accomplishing an organizational task (Pentland, Feldman, Becker, & Liu, 2012)—or events within a project or routine (Obstfeld, 2012; Pentland, Recker, & Wyner, 2017). We apply this lens in the context of a new product development project, specifically video game development. When paths are repetitive and recognizable, they represent performances of a routine (Feldman & Pentland, 2003; Obstfeld, 2012). In the project we studied, some paths were repetitive and recognizable but there was also constant change, making it a good context to study routine dynamics. We use evidence from this project to theorize about a central problem in routine dynamics: What drives a pattern of action to become more or less varied?

We build on the concept of patterning (Danner-Schröder & Geiger, 2016; Feldman, 2016a; Turner & Rindova, 2018) as a way to describe routine dynamics. We conceptualize patterning as the formation of new paths and the dissolution of old paths in the narrative network that describes the routine (Pentland & Feldman, 2007). The general approach is analogous

to established models of social network dynamics (Snijders, 2001; Snijders, van de Bunt, & Steglich, 2010), but instead of examining ties between a fixed set of *actors*, we trace paths between a constantly changing set of *actions*. We find that action patterns change dramatically over time depending on project needs, and explain generic mechanisms that lead to more (or fewer) paths being enacted in the narrative network. While these mechanisms relate to Van de Ven and Poole's (1995) classic typology of change motors, we propose that patterning can be seen as a novel motor of change for routine dynamics.

A path-based focus is not a minor methodological twist. It goes hand in glove with a theoretical perspective called "strong" process theory (Hernes, 2014; Langley & Tsoukas, 2016a; Tsoukas & Chia, 2002). Strong process theory offers a radical, process-centric ontology of the social world. Tracing the formation and dissolution of paths over time provides a concrete way to operationalize strong process theory in empirical research on routine dynamics. Our path-based approach offers a new way of seeing and measuring how the patterns of action in a routine have changed. This allows us to describe and theorize about the mechanisms that drive routine dynamics.

THEORY

The philosophical roots of strong process theory can be traced to Whitehead (1929/1978), James (1909/1996), Mead (1934/1962), and Dewey (1938/2008), and more recently the work of Chia (2016), Hernes (2014), Rescher (1996), Shotter (2006), and others (see Langley & Tsoukas, 2016b). The basic insight is simple. As Weick (1979: 95) observed, "organizations are grounded in interlocked behaviors rather than interlocked people." Putting actions in the foreground, rather than actors, aligns with the view that the social world is a continually unfolding process (Strauss, 1993; Tsoukas & Chia, 2002). Thus, the "dynamic, unfolding process becomes the primary unit of analysis rather than the constituent elements themselves" (Emirbayer & Mische, 1998: 287). This strong process view has been widely adopted in research on routine dynamics (Howard-Grenville & Rerup, 2017). In the following sections, we review the routine dynamics literature and explain how paths can provide a new way of seeing and characterizing the dynamics of organizational routines.

Routine Dynamics

Routine dynamics focuses on the stability and change of organizational routines from a processual

perspective (Feldman et al., 2016). Routines are repetitive, but because each performance of a routine unfolds over time it can always unfold in a new direction (Feldman & Pentland, 2003). While there are many factors that help routines "stay on track" (Schulz, 2008), routines are not constrained to follow predefined paths (Feldman, 2000; Feldman & Pentland, 2003; Feldman et al., 2016). Future paths are influenced by past paths, but not determined by them. Furthermore, as Feldman et al. (2016) pointed out, some routines are not very routine: they embody an enormous number of possible paths (Hærem, Pentland, & Miller, 2015; Pentland, Hærem, & Hillison, 2010). All routines exhibit what (Cohen, 2007) called "pattern-invariance," but some are more varied than others and, at the same time, the patterns may be changing.

This points to a central puzzle in routine dynamics: What makes a pattern of action more or less varied? In routine dynamics, variety enables change (Feldman, 2016a; Pentland, Liu, Kremser, & Hærem, Forthcoming). A pattern of action that is more varied encompasses more paths, with more possibilities for divergence or change. A pattern of action that is less varied encompasses fewer paths, with fewer possibilities for divergence or change. However, the theoretical problem of what drives patterning is not explained: Why do patterns of action stay the same or change over time? There is also the methodological problem of seeing and quantifying pattern in variety (Cohen, 2007). We cannot research this phenomenon if we cannot see it.

The growing body of field research on routine dynamics has focused on explanations of stability and change. It has elaborated on the concept of endogenous change as theorized by Feldman and Pentland (2003), and pointed to the importance of exogenous factors as well. For example, in their study of compliance routines in oil exploration, Bertels, Howard-Grenville, and Pek (2016) showed how routines can be shielded from and shored up against external interventions. The routines remain stable, although this stability requires effort and continual maintenance. In contrast, in their study of NASA's implementation of an enterprise information system, Berente, Lyytinen, Yoo, and King (2016) showed that routines can change through unanticipated local adaptation. As Barley (1986) observed when computerized tomography (CT) scanners were introduced into radiology departments, Berente et al. (2016) found that new technology can lead to new patterns of interaction in a workplace. At the organizational level, Rerup and Feldman (2011) demonstrated how organizational routines coevolve with organizational schema through different types of "trials" and "errors."

The formation of new routines has also been a topic of considerable interest. For example, in the context of video game development, Cohendet and Simon (2016) described a process of forming new routines through deliberately breaking, partitioning, and recombining aspects from different routines in response to an organizational disruption that required them to shift from efficiency to make room for creativity. Deken, Carlile, Berends, and Lauche (2016) showed how flexing, stretching, and inventing generated novel actions and outcomes in an automotive supplier that was developing a new line of information-based services. Meetings (Aroles & McLean, 2016) and spaces (Bucher & Langley, 2016) provide opportunities for questioning, reflection, and thought experiments as participants work out new routines (Dittrich, Guérard, & Seidl, 2016).

This fieldwork provides evidence that routines do, in fact, change over time in systematic ways, and has led to a refined understanding of factors driving stability and change in routines. Feldman et al. (2016) noted that these field studies have generally employed situated actions as the unit of *observation*, and patterns of action as the unit of *analysis*. Ironically, the unit of analysis (the pattern of interdependent action that makes up the routine) has been less visible. The literature on routine dynamics has theorized about patterns of action, often without measuring or visualizing those patterns (Feldman, 2016b).

Feldman (2016a: 38) suggested that one way forward is to focus on the “inseparability or mutual constitution of actions and patterning.” Patterning exemplifies the process of dynamic unfolding described by Emirbayer and Mische (1998), because routines are performed one step at a time. Step by step, situated actions enact recognizable paths. Paths represent a missing link between situated actions and repetitive patterns. By tracing paths, we can begin to connect actions and patterns.

Routine Dynamics as Network Dynamics

In this paper, we use narrative networks to represent organizational routines and trace paths within

routines (Pentland & Feldman, 2007; Pentland et al., Forthcoming). A narrative network is unlike a social network because the nodes represent events or activities, not people. The ties (edges) in a narrative network represent sequential relations between the actions and can be interpreted as *handoffs* (Pentland, Recker, & Wyner, 2017). They could also be interpreted as organizing moves (Pentland, 1992) because they enact division of labor, hierarchy, and other organizational structures. For example, in the video game development project, there were constant handoffs between work activities and departments (e.g., art work and programming).

Technically, a narrative network is a directed graph where the weights on the edges can be used to quantify the frequency of handoffs between activities. Pentland and Liu (2017) described methods for constructing narrative networks from data collected in field research. These networks can be automatically constructed from computerized event logs or observations using software provided by Pentland, Recker, and Wyner (2015, 2016). In the context of organizational routines, the narrative network thus represents the patterning of actions in performing the routine. The differences between social networks and narrative networks are summarized in Table 1.

Network dynamics. In models of social network dynamics, changes to the network are modeled by adding and removing ties between the individuals in the network. Tie formation is driven by reciprocity (Wasserman & Faust, 1994), preferential attachment (Barabási & Albert, 1999), homophily (McPherson, Smith-Lovin, & Cook, 2001), transitivity (Davis, 1970; Holland & Leinhardt, 1977), and other features of the network. There are established models for predicting dynamics (Snijders et al., 2010) and for visualizing dynamics (Handcock, Hunter, Butts, Goodreau, & Morris, 2008; Moody, McFarland, & Bender-deMoll, 2005). Relational event models (Butts, 2008; Leenders, Contractor, & DeChurch, 2016) that predict the likelihood of a relational event (i.e., interpersonal action) between two parties provide a way to model social network dynamics in continuous time.

TABLE 1
Social Network Versus Narrative Network

	Social Network	Narrative network
Network	Represents relations among a set of people	Represents sequential relations among a set of actions
Nodes (Vertices)	Individual people	Actions or events
Ties (Edges)	Connections between people	Handoffs between actions or events
Paths	Degrees of separation (“hops”)	A possible way to perform part of a process; a recipe for action

We conceptualize narrative network dynamics in an analogous manner to social network dynamics: as the formation and dissolution of network edges. However, narrative networks and social networks are fundamentally different ways to see the social world. While social networks represent the ties between actors, narrative networks represent sequential relations between actions or events. In narrative networks, nodes are not individuals with cognition, motivation, and other personal characteristics. Consequently, the mechanisms that drive the dynamics of social networks do not apply. For example, it does not make sense for actions to be attracted to each other and become sequentially related on the basis of that attraction. Thus, to theorize about the dynamics of narrative networks, we need to start from scratch. For this purpose, we turn to the concept of network paths.

Network paths. In any kind of network, a *path* is defined as a sequence of connected nodes (West, 2001). However, the interpretation of paths is different in different kinds of networks. In a social network, a path counts the number of “hops,” or degrees of separation, between individuals in the network. The shortest path provides a measure of distance between nodes. It is an indication of connectivity between pairs of nodes and can be used to identify nodes or ties that are critical for connectivity (Freeman, 1977; Wasserman & Faust, 1994: 105).

In a narrative network, a path represents a sequence of actions that might be used to carry out part of an overall routine or process. Paths can also be considered as recipes for action or stories: they describe how a process has been or could be performed. Like any recipe or story, it is carried out one step at a time. The nodes in the network are the actions and the edges represent the movement from one action to the next, connecting those actions into paths.

Thus, we see a path as a sequence of steps enacted over time. Building on Strauss (1993), Obstfeld (2012) used the term *trajectory* to refer to the same basic idea. Obstfeld (2012: 1574) defined a trajectory as “a sequence of interdependent actions involving multiple actors.” In business process management, paths are often referred to as “traces” (Song, Günther, & Van der Aalst, 2008). While we are referring to the same concept, we prefer the term *path* because it emphasizes the graph theoretic interpretation (West, 2001).

Steps to paths to patterns. The narrative network provides a theoretical explanation of how enacting different steps influences the possible paths in a routine. When we add or remove steps (edges) from a

narrative network, it changes the set of possible paths. It creates (or removes) possible ways of getting things done. For example, if a new bus route or subway line opens (or closes), it may create (or remove) a possible path for getting to work. As a result of these changes, new paths become available and old paths become unavailable. Each possible path contributes to the overall pattern.

In general, adding actions (nodes) or handoffs (edges) will tend to *increase* the number of paths. Removing actions (nodes) or handoffs (edges) will tend to *decrease* the number of paths. These relationships are not hypotheses; they are based on mathematical properties of directed graphs. The question for organizational research is: What mechanisms drive these dynamics?

METHODS

We conducted a field study of a video game development project team that involved being “in the flow” to capture longitudinal data through observations, interviews, and archival materials. In-depth fieldwork provided the fine-grained detail necessary to bring the phenomena to life (Feldman et al., 2016; Jarzabkowski, Lê, & Spee, 2016).

Research Setting

The setting for our study is a project team, ProjectBQ, at a video game development studio, GameSG (both pseudonyms), based in a mid-Atlantic city in the United States. During the period of data collection, GameSG was a 10-year-old studio that employed approximately 60 employees, mostly under 30 years of age, with expertise in software engineering, game design, and technical art. Prior development projects at GameSG included games on various platforms (e.g., mobile phones, standalone entertainment systems, TV plug-in games, Internet browser games) for a wide spectrum of clients that included video game publishers, media conglomerates, theme parks, and a startup toy company.

Project teams in GameSG were usually composed of members with expertise in one of the following skill sets—game design, software engineering, technical art, script writing, animation, sound composition, and project management. The composition of team members in ProjectBQ was typical in this regard. The team was led by a core group of functional “leads” consisting of the producer, a lead designer, a technical lead, and an art lead. Each lead was responsible for coordinating work in that

functional domain and acting as a gatekeeper for the quality of work produced. Project leads were also directly involved on high-level decisions about the design and functionality of the game. The producer managed deadlines, the pace of work, and access to resources for the team. They played a boundary-spanning role between the team and other stakeholders, such as GameSG management, other project teams, and the client. The team size for ProjectBQ ranged from eight to 15 developers over a 14-month period.

ProjectBQ was funded by a nonprofit with the goal of promoting anti-drug messages through unstructured learning methods. The project was a “serious” game intended to teach teenagers about resisting peer pressure in high-risk situations (e.g., substance abuse, risky behavior). The game was themed as a fantasy game where the hero protagonist is a mouse that is attempting to protect his tribe from the corrupting influence of the villain antagonist. Players progressed in the game by visiting new worlds to battle enemies. Battles were turn-based and were won by whether the player picked the right move that would best counter the one chosen by the computer. Although the game had “fantasy characters,” players had to make decisions based on real-world situations. As described by Producer1,

It is not direct messages saying, “Don’t do drugs.” What it’s saying is, “Here are some situations that you’re not going to be comfortable with in real life. Here are responses and ways in which you can handle those situations without feeling like a nerd or an outcast, or like you’re going to lose your friends or things like that. (Producer1)

ProjectBQ was typical of the game development projects at GameSG in that the stages of development followed a standard sequence of a preproduction, production, and refinement. The preproduction stage involved testing out ideas for the game with the goal of finalizing game design. The production stage involved building the actual game. Finally, the refinement stage involved fixing software bugs and improving on the playability of the game. Despite following this standard sequence of development, ProjectBQ team members were more frustrated than usual about the frequent design changes. The project was over scoped and behind schedule. These issues manifested in a few notable incidents during the project: the project lead (who was also one of the tech leads) was replaced with a codesigner, the lead designer was fired from the studio, and the studio head had to become personally involved in redesigning

the game halfway through. As the project developed, team members reported losing interest in the project and were unhappy at having to work overtime on a game they did not find fun at all. The game was eventually built and delivered to the client, albeit behind schedule. Despite the negativity in the development process, the game was found to have moderate success in improving adolescent players’ ability to identify pressuring situations, as well as to recognize and practice healthy responses. The game was also a finalist for several gaming awards and was rated 4.2 stars on Google Play and 4 stars on iTunes, out of a possible 5 stars.

We picked video game development as an exemplary setting for studying routine dynamics because it is a collective task that is ambiguous and emergent: there are an endless number of possibilities for combining elements to create a game. Video games are an interactive virtual experience produced by a computer program onto a display device that people engage in for entertainment. Although games are also used in more “serious” settings, such as education and training simulations, there is always an element of interactivity and engagement with the player. However, how this interactivity and engagement manifests in the context of the game is rarely obvious at the outset of game development (Cohendet & Simon, 2016).

These characteristics of video game development can be considered a type of creative project (Obstfeld, 2012). Creative projects consist of an emergent trajectory of interdependent action initiated and orchestrated by multiple actors to introduce change into a social context. The nature of these departures could be in the form of new elements, or new linkages between familiar elements. The ambiguous means and ends of creative projects imply that “repetition is not a guide on what to do next” (Obstfeld, 2012: 1571) as the trajectory of action required to create the video game does not follow a set plan. On a continuum of routine and nonroutine actions, ProjectBQ is clearly at the nonroutine end of the continuum (Adler & Obstfeld, 2007; Obstfeld, 2012).

Data Collection

Our research design incorporated data from archival materials, nonparticipant observation, and interviews. Data were collected over 15 months as part of a longer two-year study on the routines in video game development. The ProjectBQ team used a software project management approach called

“scrum” (Cohendet & Simon, 2016; Sutherland & Sutherland, 2014). Scrum involved breaking down the project into three-week “sprints.” Before each sprint, the team would decide on their collective goals and individual tasks for the next sprint. The sprint consisted of short daily meetings, lasting no more than 15 minutes, where members updated the team on the progress of their individual tasks. At the end of the sprint, the team would meet to review the progress on the team’s goals and set their goals for the next sprint. This cycle continued for the entire duration of the project.

The primary document we relied on to construct networks of action patterns were “scrum sheets”—archives of task schedules that contained logs of tasks assigned to each individual. These documents were updated daily by the team, and daily versions of these documents were downloaded between May 2011 and February 2012 ($n = 122$). As an archival source, the scrum sheets are particularly suitable for capturing chronologies of actions over long periods of time (Langley, Smallman, Tsoukas, & Van de Ven, 2013).

The scrum sheets were used to create a database of tasks, the “story” or goal that it meant to accomplish, the actors associated with the tasks, and when the task started and ended. A “difference report” was created for each day by comparing scrum sheets with the most recent version to identify which tasks were added or removed, and the progress made on the task. From these daily difference reports, a list of actions was created ($n = 2,803$). Starting and ending dates for each task were also extracted from the difference reports. Tasks without a start and end date were removed as these tasks were not acted upon, resulting in final list of 2,428 tasks. These actions were then grouped by stories and sequenced according to the following order: (1) when the task ended, (2) when it was started, and (3) the order in which the action was added to the database. The last criterion was necessary to determine the ordering of actions that shared similar start dates and end dates.

Between May 2011 to August 2012, the first author was a nonparticipant observer on ProjectBQ. These observations included team meetings ($n = 39$), client meetings ($n = 7$), and play test sessions ($n = 4$). Team meetings included daily 15-minute “scrum” meetings ($n = 29$) where team members met to schedule and coordinate their tasks for the day, retrospective meetings where they reviewed work processes ($n = 2$), and general discussions about the project ($n = 8$). During these meetings, notes were taken about the purpose of the meeting, what was said and by whom,

and the author’s impressions of what transpired during the meeting.

In addition to data from observations, both *ad hoc* informal ($n = 11$) and formal semi-structured interviews ($n = 4$) were conducted with team members. The informal interviews focused on getting status updates on the project, while formal semi-structured interviews were about 60 minutes long and focused on gaining an in-depth understanding of specific episodes during the project. Interviews were conducted with the producer, the two tech leads, the art lead, a designer, and a software engineer. Archival materials such as project schedules, planning documents, meeting notes, and budgets were also accessed and referenced to establish rich insights into the events surrounding the actions taken by the team.

Data Analysis

In keeping with our goal of seeing and theorizing about patterning as it was enacted over the course of the project, we analyzed data chronologically as a narrative. The data analysis consisted of three main steps: (1) constructing a series of narrative networks that represent patterns of action throughout the project, (2) computing the properties of each network, and (3) constructing a project narrative to interpret and theorize about the dynamics of those patterns.

Constructing narrative networks. Narrative networks were constructed in the following steps: (1) code the data into sequences that can be used to construct narrative networks, (2) bracket the data into windows of analysis that correspond to project sprints, and (3) construct and visualize the networks through a software application called ThreadNet (Pentland et al., 2015, 2016).

The first step, coding the data into sequences, required coding the final list of 2,428 activities from the scrum sheets according to the actions and roles involved in each activity. We used a constant comparative process (Glaser & Strauss, 1967) to develop task categories with the help of two research assistants. Categories were developed by iterating between the first author’s familiarity with the context, field notes, and other archival documents to understand the intent of the task. This process involved forming initial clusters of tasks to minimize differences within clusters while maximizing differences between clusters. An initial set of categories was then developed from these clusters. New tasks were compared with earlier tasks in the same category. If a

newly categorized task appeared to be different from other tasks in the same category, this would be reconciled by attempting to refine the definitions and properties of these categories to accommodate the new data. This process of constantly comparing new data with existing codes was continued until a level of stability was reached. From 12 initial categories, the list was ultimately reduced to the following six categories: Administration, Experimenting, Building, Revision, Refinement, and Testing (Table 2). Figure 1 shows the distribution of these categories over time.

Roles were coded in a similar approach to coding actions. The primary actor responsible for each task in the database was categorized into an organizational role by the first author based on the researcher's familiarity with the research setting. These roles were Design, Art, Tech, and Analytics (Figure 2). Together with the actions, these roles define the possible actions in the narrative network. The six roles and six task categories meant that there were potentially 36 unique role-task categories. These 36 categories were applied to the

2,428 time-ordered events gathered from the scrum sheets to create a set of 159 coded sequences. These coded sequences become the input for creating a series of narrative networks for the project as it progressed.

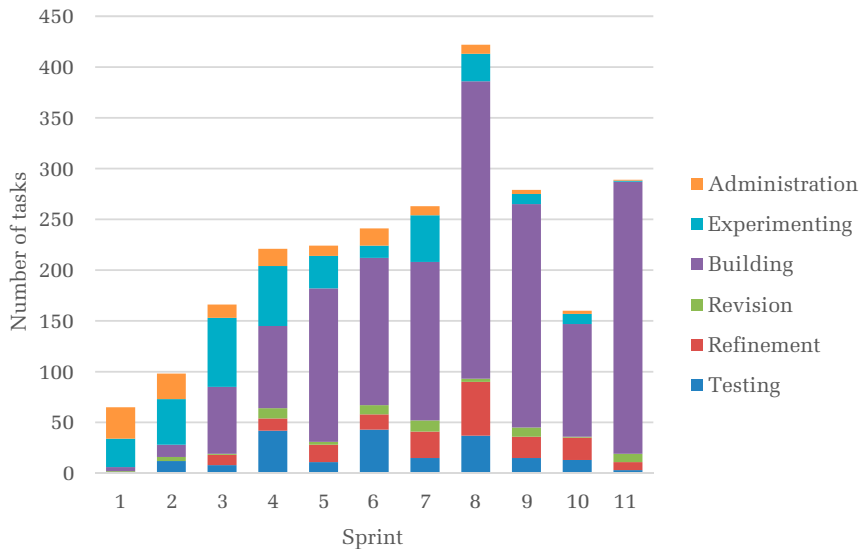
The second step involved bracketing narrative networks into windows of analysis. ProjectBQ was implemented using an agile software development methodology (Moe, Dingsøyr, & Dybå, 2010; Sutherland & Sutherland, 2014), which meant that the project was divided into three-week long phases, again called "sprints." For our analysis, we bracketed the data (Langley, 1999) into three-week windows that corresponded with the dates for each sprint.

The third step involved constructing and visualizing the networks through ThreadNet (Pentland et al., 2015, 2016). ThreadNet was used to convert the coded sequences into a narrative network for each sprint. The application traces the coded sequences of action to create networks. Each kind of coded action becomes a node in the network. Adjacent pairs of coded actions become edges in the

TABLE 2
Definition of Task Categories

Category: Final	Category: Round 1	Definition
Administration	Administration	Activities that involve planning, organization, coordination, communication with internal or external parties.
Experimenting	Experimenting	Activities associated with learning, discovery, building experience or knowledge, addressing unanswered questions.
	Conceptualization	Activities associated with defining the form of team output. Includes definition of interrelationships between components of team output, how output fits with client's other activities (e.g., marketing). Manifests as transitional output or boundary objects.
Building	Building	Activities directly associated with producing assets.
	Integration	Activities associated with combining different parts of the team output (e.g., art assets).
Revision	Revision	Activities associated with rebuilding, reimplementing, redesigning, or rewriting. Adjustments made to core aspects of output (e.g., code, model, animation) in terms of the relationship between parts. If the relationship between A and B can be specified in an equation, this will involve changes to variables in the relationship, rather than the absolute value of the variables.
Refinement	Refinement	Activities associated with adjusting parameter values of output.
	Fix	Activities associated with rectifying errors. Closely related to "Tweak," but difference here is that the adjustment is made to some part of output that is broken, or not working as it should. Words like "correct," "error." Result or outcome is unintended.
Testing	Review	Reviewing work before release.
	Testing	Activities directly associated with enacting playtests. Different from QA tests, which check technical integrity of output.
	Feedback	Activities related to obtaining or aggregating feedback from playtests or metrics, by clients or users. Related to the event of obtaining feedback.
	Quality Assurance (QA)	Activities that involve testing for bugs, errors, or edge cases. Different from playtests.

FIGURE 1
Frequency of Task Types Across Sprints



network. Although this procedure can be performed manually, the software is faster and less error prone.

Computing properties of narrative networks.

Once the networks for each sprint were constructed, we computed their properties. The two basic properties that define any network are (1) the list of nodes and (2) the list of edges (West, 2001).

For our purposes, we simply needed to count the number of nodes and edges in each graph. These counts are provided automatically in ThreadNet (and other network analysis tools), and are shown in Table 3.

To estimate the number of paths in the network, we used a simple formula based on McCabe’s (1976) concept of cyclomatic complexity:

FIGURE 2
Frequency of Roles Performing a Task Across Sprints

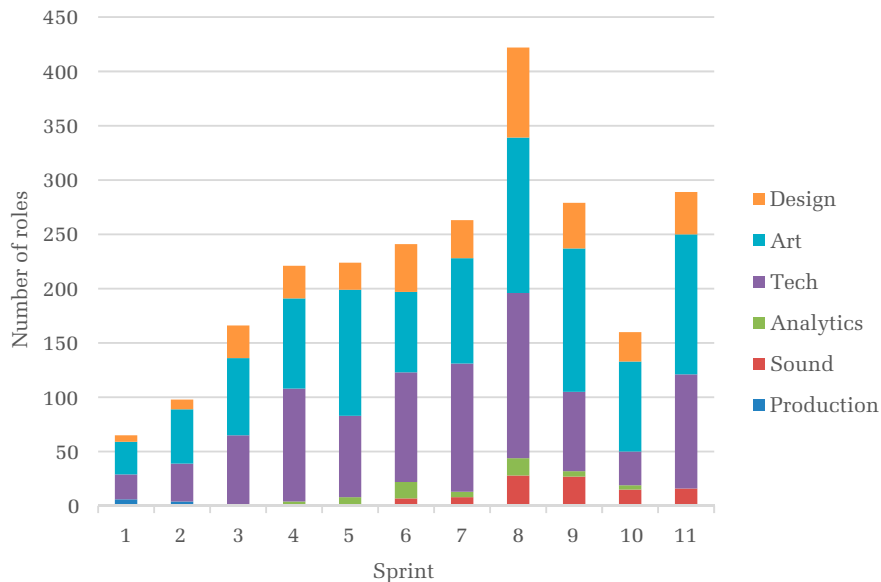


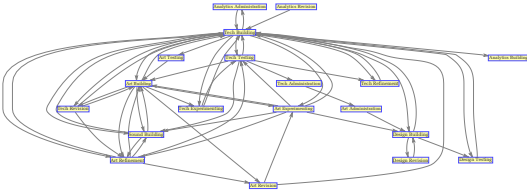
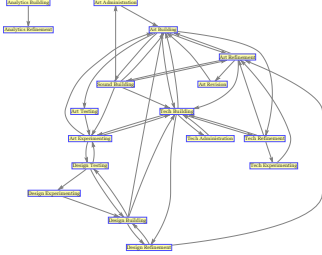
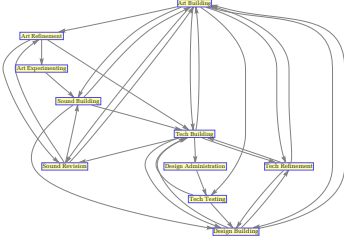
TABLE 3
Summary of Mechanisms, Complexity Index, Actions, Handoffs, and Graph Diagrams

Sprint	Graph	Complexity Index	Actions (Nodes)	Handoffs (Edges)	Paths Added or Dropped	Mechanisms
1		0.16	9	19	n.a.	Performance
2		1.84	13	43	68	Performance
3		2.56	14	55	294	Performance Revision
4		4.64	17	87	43289	Performance Revision

TABLE 3
(Continued)

Sprint	Graph	Complexity Index	Actions (Nodes)	Handoffs (Edges)	Paths Added or Dropped	Mechanisms
5		1.52	17	47	-43618	Performance (reduction)
6		3.92	21	83	8285	Performance Revision Delay Motivation
7		4.32	21	86	12575	Performance Revision Delay Motivation
8		5.76	21	106	554547	Performance Revision Delay Motivation

**TABLE 3
(Continued)**

Sprint	Graph	Complexity Index	Actions (Nodes)	Handoffs (Edges)	Paths Added or Dropped	Mechanisms
9		3.12	18	66	-574122	Cut-back Performance
10		2.0	16	51	-1218	Cut-back Performance
11		1.74	10	38	-45	Cut-back Performance

$$Estimated\ paths = 10^{0.08 * (Edges - Nodes + 1)} \quad (1)$$

Stated in English, the estimated number of paths in a network is an exponential function of the difference between the number of edges and the number of nodes. For a given number of nodes, increasing the number of edges will increase the estimated number of paths. The constant (0.08) is derived empirically by fitting this equation to thousands of simulated networks with a known number of paths. The derivation, validation, and limitations of this formula are provided in Appendices A and B. The complexity index (Hærem et al., 2015) is computed as a logarithmic function of the number of estimated paths.

Constructing the overall project narrative. We constructed a timeline of events from interviews with informants. These interviews were professionally transcribed and analyzed using nVivo software to identify

periods, major events, and the critical actors associated with the temporal unfolding of the project (Langley, 1999; Pentland, 1999). We drew on the first author’s observations of the project team to validate our timeline of the project. Each observational event was dated and summarized. We then compared the events provided by informants with these observations to validate the timeline.

To create a more detailed narrative, we augmented the basic project timeline by iterating between the interviews and the observations, with an emphasis on the contextual circumstances surrounding interpretations of why events occurred, individual thoughts and feelings in response to actors and incidents, and histories. This narrative provided in-depth insights into the unfolding project that extended temporally across the past and into the future, and across actors that included individuals, the team, and external stakeholders.

FINDINGS

We report our findings in two parts. In the first part, we describe *four phases of patterning in ProjectBQ*. During each phase, the number of paths in the narrative network increased or decreased dramatically. In the second part, we combine our qualitative data about the project with quantitative metrics about the narrative network to theorize about the *mechanisms that drive routine dynamics*.

Four Phases of Patterning in ProjectBQ

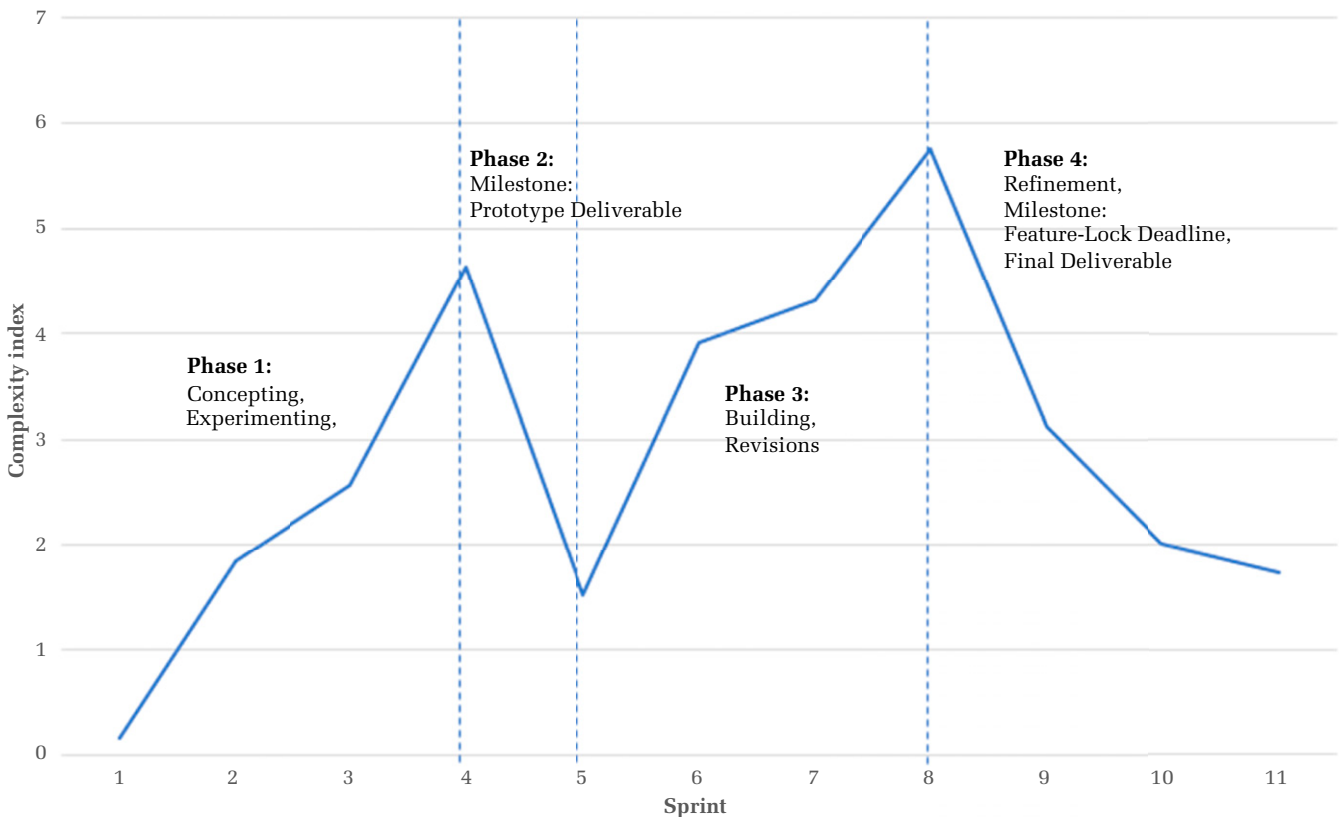
As a creative project, ProjectBQ involved a lot of change. Figure 3 shows how the complexity of the project (indexed by the number of paths) changed over time. Table 3 shows the narrative network for each sprint, plus the number of distinct nodes or edges and the number of paths added or removed from one sprint to the next. Table 3 also details the mechanisms that drive dynamics, which are explained in the next section. Here, we discuss the

project in four phases of patterning that correspond to distinct changes within the project.

Phase one: Sprints 1 to 4 (increasing complexity). In phase one, complexity increased between Sprints 1 to 4 (adding over 43,000 paths to the network). This increase was driven by both an increase in the number of distinct actions (from nine to 17), as well as the number of distinct handoffs (from 19 to 87).

This increase can be explained by the fact that the first phase of the project consisted of concepting, prototyping, and developing the core mechanics of the game. Sprint 1 was designated as the phase to develop “Initial Concepts.” while Sprint 2 was initially designated as a “Preproduction” phase, the goal of which was for developers to rehearse the steps for producing game assets and incorporating these assets into the game to get a sense of the production schedule. Going through this process helped them to “make sure a lot of these later milestones were laid out and could be accomplished” (Art1). However, Sprint 2 was later

FIGURE 3
Complexity Index for Sprints 1 to 11



renamed as a “Production” phase. Sprint 3 was assigned to be the phase for developing “Battle prototype,” which was a core mechanic of the game. This was to be followed by a phase for developing the combat system and the game environment in Sprint 4, which was labeled the “Combat, Burrow” phase. To develop the “combat” feature of the game, the Designer needed to account for technical and aesthetic concerns, which required closer collaboration, coordination, and iteration with Tech and Art. This interdependence between developers from different functions is evident from the doubling in distinct handoffs from Sprint 2 to Sprint 4.

Phase two: Sprint 5 (decreasing complexity). In Sprint 5, complexity decreased to 1.52. Interestingly, the number of distinct actions remains the same, at 17. The decrease in complexity was driven by the decrease in handoffs ($n = 47$), which resulted in a decrease of over 43,000 possible paths. Complexity declined in Sprint 5 because a prototype was to be delivered to the client at the end of the sprint. As a result, most of the actions were building-related, as is evident from the increase in frequency of Building tasks in Sprint 5 (Figure 2).

Phase three: Sprints 6 to 8 (surge in complexity). In phase three, complexity increased between Sprints 6 to 8 (from 3.92 to 5.76). The number of distinct actions increased slightly from Sprint 5 but remained constant throughout this phase ($n = 21$). However, the number of distinct handoffs more than doubled from Sprint 5 (from 83 to 106). This seemingly minor increase in handoffs led to a dramatic addition of over 500,000 possible paths. This change is especially striking because the number of distinct actions was constant during this phase.

This enormous increase in possible paths resulted from the ProjectBQ developers working toward a “gamma build” deliverable that was due in Sprint 11. As the “feature lock” deadline was in Sprint 9, there was a flurry of activity that included both Experimentation- and Building-related actions to confirm the final features of the game in Sprint 8. We found that the increase in handoffs was due to team members iterating between downstream roles (e.g., “Sound”) and tasks (e.g., “Refinement”) and upstream roles (e.g., “Design”) and tasks (e.g., “Experimenting”).

Phase four: Sprints 9 to 11 (decline in complexity). In phase four, complexity decreased between Sprints 9 to 11 (from 3.12 to 1.74). This decrease in complexity was caused by a decrease in both distinct actions (from 18 to 10) and distinct handoffs (from 66

to 38). The number of possible paths dropped off by over 500,000, mostly in Sprint 9.

In phase four, there was a decline in both distinct actions and handoffs because the feature lock deadline in Sprint 9 meant that no more changes to the design could be made. Hence, the remaining actions were mostly Building related. There were no longer major design changes that required developers to iterate between experimenting and building, or between functions.

Mechanisms that Drive Routine Dynamics

In the second part of our findings, we draw on our results to identify mechanisms that drive the complexity of routine dynamics through the addition (or removal) of actions and handoffs from the network. These mechanisms operate to varying degrees throughout the performance of the project. To identify these mechanisms, we draw on causal loop diagramming methods, which are commonly used in system dynamics research to articulate process theories (e.g., Rudolph, Morrison, & Carroll, 2009; Strike & Rerup, 2016), to unpack how events unfolded in ProjectBQ. We identify a total of six mechanisms that directly affect the complexity of routine dynamics: reinforcement loop, performance loop, revision loop, delay loop, cut-back loop, and motivation loop.

Reinforcement loop. Reinforcement through repetition is one of the basic mechanisms of stability in routines (Cohen & Bacdayan, 1994; Schulz, 2008). In ProjectBQ, we found that the frequency of a handoff in one sprint was positively related to the tendency for that handoff to occur in the next sprint. If a handoff appeared more than once in a given sprint, there was a 55% chance it would appear in the next sprint. If a handoff appeared more than five times in a given sprint, the chance of it appearing in the next sprint increased to 95%. These findings thus provide evidence of stable, repetitive patterns of action even within the context of a creative project. By itself, repetition tends to reduce complexity because, in a routine with many thousands of possible paths, stronger paths get reinforced and weaker paths are forgotten (Pentland et al., Forthcoming). Conversely, greater complexity (more paths) reduces the chances that a particular path will be repeated. We label this relationship between repetitive patterns of action and complexity as the reinforcement loop (Figure 4a).

Performance loop. Figure 4b shows the set of relationships in the performance loop that drive

actions and handoffs. The existence of an output quality gap—the gap between the client’s requirements and the overall quality of the project team’s output—instigates the developers to act to narrow this gap. This mechanism was evident throughout the project, but manifested in different ways at different sprints.

At the beginning of the project, particularly in Sprints 1 and 2, actions were taken to help designers to understand the game mechanics and make decisions about the features and functionalities that the game would have. Producer1 explained the process at this stage as follows:

[The designer] felt that we should have a preproduction phase. Your designer needs a preproduction process because they need to figure out what the game is and then they need to start designing it before the tech people come in and start building it. You can’t build something that hasn’t been figured out yet. (Producer1)

The goal of the preproduction phase was for developers to rehearse the steps for producing game assets and incorporating these assets into the game to get a sense of the production schedule. Going through this process helped them to “make sure a lot of these later milestones were laid out and could be accomplished” (Art1). A large proportion of actions in Sprints 1 and 2 thus consisted largely of Experimentation actions performed by the core group of eight developers.

In Sprint 3, the team’s headcount increased because the Tech lead lobbied senior management to bring on more developers to the team sooner. This decision was made due to concerns that the project had been over scoped, which would hurt their ability to meet project deadlines. With the increase in headcount, there was also a corresponding increase in the total frequency of actions from 65 in Sprint 2 to 98 in Sprint 3 as developers “ramped up” and moved into the Production phase to build the game, even while continuing to experiment with different ideas. The increase in actions and handoffs enabled the team to work toward their first major project milestone—to deliver a playable prototype to the client at the end of Sprint 5.

We have thus far explained how the performance loop increased actions and handoffs. However, this mechanism could also reduce actions and handoffs when expectations for output quality were low, such as in Sprint 5. In Sprint 5, the complexity index decreased from 4.64 in Sprint 4 to 1.56. This decrease in complexity index was due to there

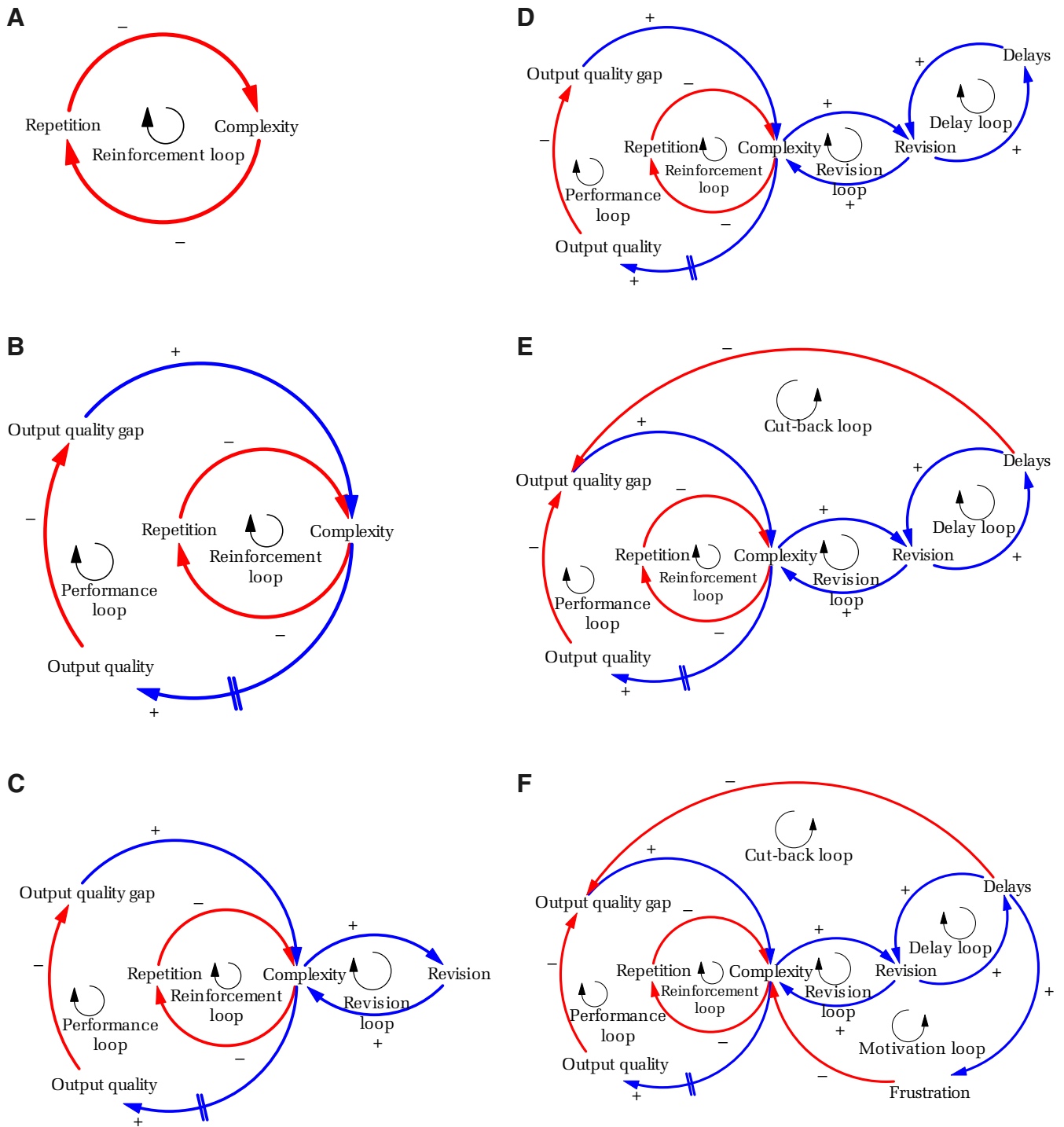
being fewer handoffs, since the frequency of actions was approximately similar in both Sprints 4 ($n = 221$) and 5 ($n = 224$). The reason why there were fewer handoffs in Sprint 5 is because the developers were focused on completing the prototype at the end of Sprint 5. After they had iterated a design that they thought was good enough to meet the client’s expectations for this milestone, the team focused on Building and Refining actions to build the prototype. Thus, the proportion of Building actions increased from 36.7% in Sprint 4 to 67.4% in Sprint 5. Of note were the decreases in Testing actions, from 19.0% in Sprint 4 to 4.9% in Sprint 5, and decreases in Experimenting actions, from 26.7% in Sprint 4 to 14.3% in Sprint 5. Furthermore, since quality expectations for prototypes were lower, developers did not need to iterate between functions and revise their work as frequently, which explains the fewer handoffs and paths. This relationship between output quality gap, actions and pathways, and output quality partially explains the dynamics of complexity, which increased between Sprints 1 to 4, then plunged in Sprint 5.

In the second half of the project, from Sprint 6 onwards, the ProjectBQ team had a new milestone to deliver a “gamma build” of the game in Sprint 11. New actions and handoffs were undertaken to develop the game to meet this milestone, which led to the increase in complexity index between Sprints 6 to 8.

After the feature lock deadline in Sprint 9, the game could no longer be improved by adding or modifying game features. Consequently, this narrowed the output quality gap by reducing quality expectations to refining or “polishing” the features that were already in place. The narrowing of the output gap due to the feature lock deadline instigated a corresponding shift toward Building and Refining actions, with fewer pathways across roles, which led to the decline in complexity from Sprints 9 to 11.

Revision loop. The Performance loop also intersected with other mechanisms, one of which was the Revision loop. As more components of the game were completed, developers could playtest the game and learn about which features of the game to change, add, or remove. This feedback triggered revisions to the design, which led to more actions and handoffs between roles to accomplish, creating a positive feedback cycle that we label the “Revision loop” (Figure 4c).

FIGURE 4
Mechanisms Driving the Complexity of Routine Dynamics



The Revision loop was evident in Sprints 3 and 4 of ProjectBQ, where complexity increased from 2.68 to 4.70. By Sprint 3, the team had completed an early prototype (the “Gold Spike”) and had gone through

the process of incorporating some graphical assets into the game. Going through this production process made them aware of constraints they could not have predicted before. As Artist1 explained,

As we got more work done, we realized that this design wasn't working or this spec needed to change, which forced a rewrite of tech. It happened a lot with UI [user interface] and it happened a lot with some of the other core mechanics, like the Burrow and Combat. (Artist1)

For example, they discovered that animated movements were too "jerky." The team narrowed down their options to either reducing the size of art assets or redesigning the combat system. While reducing the size of graphics was much quicker than redesigning the game, it would also reduce its quality. To figure this out, the team first experimented with reducing graphical quality but later realized that they would have to change the combat system from "three versus three" to "one versus one." Thus, exploring these options involved several iterations between Art, Tech, and Design, which was reflected in the high number of cross-functional handoffs in the "Combat" story in Sprints 3 and 4. The iterative process also led to handoffs between actions at different stages of development. In Sprints 3 and 4, some parts of the Combat story were in the early stages of experimenting, while others were in the later stages of testing and revision. An example of an experimenting task that Design was assigned to in the Combat story was "Influences for Combat" (Sprint 4); while an example of a later stage testing task for Tech was "2nd pass on enemy AI" (Sprint 4).

Delay loop. The Delay loop is a positive feedback loop that indirectly affects actions and handoffs through revisions. In ProjectBQ, the frequent revisions to game design and features slowed down the project team's progress. Developers thus had less time to implement features that they had originally planned for, resulting in further revisions that increased actions and handoffs.

In revising the combat system in Sprint 3, for example, "changes to [the] core mechanic mess[ed] up productivity" as the requirements for many related features "changed drastically," resulting in "rework [ing] some stuff [they] had done before" or "rewrit [ing] code from scratch" (Artist1). Revisions thus threw the production schedule for the entire project off track—not only did they have less time to accomplish their remaining tasks, but there was also more work due to the revisions.

Another example of revisions causing delays was evident between Sprints 6 to 8. ProjectBQ was

characterized by frequent revisions where "there was a new idea or new situation that will then change" (Artist 1) roughly every two weeks. Consistent with this claim, we found evidence of an iterative process in these sprints from the presence of upstream roles (e.g., Design) and actions (e.g., Experimenting) performed together with further downstream roles (e.g., Sound) and actions (e.g., Refinement) during these sprints. By then, the team was already behind schedule. Sprint 6 was intended to be the phase in which they developed the social elements of the game, and was labeled "Global quest, friends list, bring friends on missions, analytics, tutorials." However, the list of tasks was still dominated by those for "Combat," "Missions," and "Burrow," which were goals for Sprints 4 and 5.

Not only did frequent revisions cause delays by slowing down the completion of goals, but they also caused delays because the frequent revisions led developers to intentionally leave their tasks uncompleted in anticipation of further revisions. As described by Artist1,

The guys get to a point where Art wouldn't actually be making any final art for anything because we weren't sure [about] spending that time. Let's say that it's going to take you 10 hours to make a final piece of art today. Well guess what? No one's ever going to get more than five hours at any task, because we don't know what's going to get cut. If you have 20 things you need to do, instead of spending 10 hours on each of those tasks, we're going to go through all of that for five hours. Hopefully, we'll have something to show for [it]. (Artist1)

The frequent revisions created the expectation that more changes were forthcoming. This expectation, coupled with having "20 things you need to do," led to a more cautious approach where tasks would only be partially completed to minimize any loss in time. While this approach might have saved time for each individual, it slowed down the team's progress further because instead of completing a task on schedule, tasks were completed only when they became a high priority, which was usually when they were behind schedule and urgently needed to be completed. As a result of these delays, the main production phase was extended by four sprints in Sprint 6, and the overall schedule was extended by two sprints, which Producer1 reported to be the first of multiple extensions over the course of the project.

Just as revisions caused delays, we found that delays also instigated more revisions. In ProjectBQ,

the team adapted to having less time by “chopping” certain features that could not be completed in the time remaining (Tech1). At the same time, however, when a feature was simplified (i.e., such as switching from three versus three to one versus one combat) the game became “not as exciting” and the developers felt compelled to “respond by making other things more exciting, more engaging” (Artist1). Thus, even as the project scope was reduced, new features had to be redesigned, which increased actions and handoffs. Revisions and delays were therefore engaged in a positive feedback cycle, which we label the “Delay loop” (Figure 4d).

Cut-back loop. The cut-back loop is a balancing feedback loop between revisions and delays that indirectly reduces actions and handoffs through the output quality gap. Because of the high degree of interdependence between components, revisions to one component led to delays that spilled over to other components and eventually slowed the progress of the entire project. For example, the November 10 meeting in Sprint 9 shows how Design1 was blocked by Tech on the “power scripting” tasks. The Tech team could not proceed because they were themselves blocked on a number of their tasks. One of the reasons for their being blocked was that Tech3 was unable to start work until Design decided how players would “level up.” However, since Design1 was faced with higher-priority tasks, he was not able to decide on the “level up” features yet. Overall, we see that developers were entangled in an intricate web of interdependence, such that delays in one component had a domino effect on the overall rate of progress. These delays cumulated until a critical point where the team ran out of time. By then, developers no longer had time to iterate and refine their work, and were just trying to complete their tasks “in a crappy way,” or “chopping” features and reducing the scope of the project (Tech1). In both cases, there was a reduction in actions and handoffs through a reduction of the output quality gap.

These dynamics manifested in Sprints 9 to 11. A deadline for a major milestone, delivering the “gamma version,” was at the end of Sprint 11. To meet this deadline, an internal “feature lock” deadline was set at the end of the second week of Sprint 9. The feature lock deadline “froze” the build because no new features were allowed to be added after the deadline. This deadline gave assurance to the developers that there would be no more major changes to the game design, which allowed them to focus on building and refining

their work. However, it also reduced their scope for making a better game—they could not improve on core design features such as game mechanics, and could only improve the quality of the game by refining features such as fixing bugs, tidying up code, and improving on lighting and textures of graphics. Thus, the negative relationship between delays and actions created a balancing feedback loop between revisions, delays, and the output quality gap, which we call the “cut-back loop” (Figure 4e).

Motivation loop. The final mechanism we identified from our data is the “Motivation loop,” which is a balancing feedback loop between revisions and individual motivations that reduces actions and handoffs. We found that by Sprint 6, the frequent revisions were taking a toll on developers’ morale. Tech2 described “a huge penalty in both morale and productivity.” This sentiment is reaffirmed by Tech1 in the following quote:

It hurts to hear when you work on something, and then you are told that, “This is going away. Just don’t worry about it anymore, like this is no longer part of the game.” That happens to some extent in game development, but it can happen more here. . . It was just like a double kick in the pants where all this work you did right is just getting thrown out of the window, and now we’re going to ask you to do it [again]. (Tech1)

These changes left them feeling frustrated, and led to a noticeable shift in individual motivations from wanting to make the “best game possible,” to just “get it done.” This meant iterating less frequently to refine the game and holding back ideas that could improve the user’s experience. As Tech2 mentioned, “you lose some quality and ideas that people could have brought up” when they became focused on “just cranking away.” Revisions thus escalated until a critical point where developers became frustrated and pulled back their efforts. This created a balancing reinforcing loop between revisions, frustration, and actions, which we label the “Motivation loop” (Figure 4f).

Adding to this frustration among the developers, the delays also meant that they had to work overtime for several weeks to complete the game. Even then, the project was completed two months behind schedule and without many of the initial features that had been initially planned for.

DISCUSSION

We began by asking a deceptively simple question: What makes a pattern of action more or less varied over time? Implicitly, this question points to a fundamental issue in organization theory: how do we explain stability and change? One of the central insights of routine dynamics is that stability and change are *both* dynamic and *both* require explanation (Feldman et al., 2016). Routines do not just automatically stay the same; reproducing a recognizable pattern takes effort and so does changing the pattern. The tension between stability and change is implicit in every step on every path.

To help make this tension visible, we have introduced conceptual and methodological innovations that provide a new way of seeing the link between situated actions and organized patterns of action. At its core, our approach is built around a narrative network that is continually (re)enacted by the situated actions of specific participants at particular times and places. These actions perform the paths in the network. In practical terms, people are just working, and the paths are simply ways of performing the work. In theoretical terms, they enact the continual unfolding (Emirbayer & Mische, 1998), becoming (Tsoukas & Chia, 2002), and patterning (Danner-Schröder & Geiger, 2016; Feldman, 2016a; Turner & Rindova, 2018) that constitute ProjectBQ. To understand how paths influence routine dynamics, we need to zoom out from actions to patterns (Gaskin, Berente, Lyytinen, & Yoo, 2014; Nicolini, 2009).

Zooming Out From Actions to Paths to Patterns

Field research enables a fine-grained focus on actions as a unit of observation, as we have seen in empirical studies of routine dynamics (Feldman et al., 2016). However, because participants and observers tend to see parts of routines, rather than whole routines, it has always been difficult to trace overall patterns of action involving multiple actors over time. This gap was one of the original motivations for narrative networks: to enable field researchers to piece together larger patterns from fragmented observations (Pentland & Feldman, 2007).

Routines start with situated actions (Suchman, 1987). In ProjectBQ, these are basic steps required to carry out the work: creating, writing, testing, revising, etc. Some research traditions zoom in to analyze the details of how particular actions inhabit and animate particular situations. For example,

research on affordances has often zoomed in on the detailed relations between actors, actions, and artifacts (Chemero, 2003; Volkoff & Strong, 2017).

In contrast, getting from actions to patterns involves zooming out in two distinct ways. First, we zoom out from actions to paths by paying attention to the sequence of actions along the path. In addition to asking “What happened?” we explicitly ask “What happened *next*?” In doing so, we locate each action in the context of an enacted path. The sequential relations between actions provide the forward motion that gets work done, but it is important to realize that neither participants nor observers are always able to see for themselves what happens next along a path. In contemporary organizations, the next step may happen in another part of the world.

Next, when we zoom out from paths to patterns, we locate each action in the context of a network of paths. The network summarizes enacted paths within a particular window of time. In doing so, it reveals the possible paths forward from each action, which may be many or few. The network of possible paths represents the emergent accomplishments (Feldman, 2000) and the pattern of interdependent actions that define organizational routines (Feldman & Pentland, 2003). By zooming out from actions to paths to networks, the pattern of action becomes visible. When we zoom out from individual actions to consider patterns, we see that the individual actions are not independent. This is a defining characteristic of organizational routines (Feldman & Pentland, 2003).

Placing actions in networks has important theoretical and methodological implications, such as the relationship between actual paths and possible paths. At any moment along a given path, there are possibilities for branching onto a different path. Possible paths are inferred based on actual, observed edges in the graph. Since narrative networks are usually quite sparse, the inferred possibilities constitute a tiny fraction of the paths that could be formed if the network was fully connected.

In ProjectBQ, thousands of possible paths could be inferred from the enacted paths. To express these possibilities, we aggregate actual paths into a narrative network (Pentland & Feldman, 2007). Because it aggregates alternative paths, the network includes possibilities that may never be actualized. These possibilities may be considered part of the latent structure of the routine. Since many of these paths would be considered unusual or exceptional by the participants, it would be inappropriate to equate the narrative network with the ostensive aspects of the routine. The ostensive aspects of a routine embody

normative, typical understandings that tend to guide action (Feldman & Pentland, 2003). In contrast, the narrative network embodies possibilities of paths that could be taken, as inferred from actual paths enacted.

In a simple routine, there may be only a few ways to do the work. Sparse networks with a handful of paths are easy to comprehend, but it is difficult to appreciate how quickly the number of paths can change and how large it can get. Our intuition is that more required actions adds complexity. This intuition is captured in the concept of component complexity (Wood, 1986). However, a larger number of actions is not the primary driver of complexity. For any given number of actions, adding more edges (more paths) will increase complexity exponentially (Hårem et al., 2015). This increase can only be estimated by counting the edges (pairs of actions) that occur along the enacted paths. The methodological move from actions to paths to networks enables an entirely new way of seeing task complexity. In ProjectBQ, we observed orders-of-magnitude changes in the number of paths from one sprint to the next, *even though the number of actions was the same*. This descriptive finding adds urgency to our research question: What drives these dramatic changes?

Patterning as a Motor of Routine Dynamics

In our study, we identify six causal loops that influence the number of possible paths in the project from sprint to sprint. Many of these are specific to the world of video game development, so we are hesitant to generalize too broadly. However, if we step back from the particulars of ProjectBQ, we can see that generic factors, such as cost, quality, and deadlines, can influence action patterns. We can interpret these causal loops in terms of Van de Ven and Poole's (1995) classic typology of change motors: life cycle, teleologic, dialectic, and evolutionary. However, *patterning* provides a novel motor of change that is closer to the phenomena we observed and a better theoretical fit for routine dynamics. To see how this is so, first consider the traditional motors of change.

Lifecycle. The lifecycle motor depicts change as a progression through a prescribed sequence of stages regulated by an underlying logic, program, or code. In ProjectBQ, we can see the whole project as a lifecycle, from conception to final deliverable. Within the project, the three-week sprints can be interpreted from a lifecycle perspective as well, since each sprint has a repetitive structure. Lifecycles provide an excellent explanation for routine repetition, but they

have not featured prominently as drivers of routine dynamics.

Teleology. The teleological motor explains change as a goal-directed process undertaken by an entity that involves a cycle of setting, implementing, evaluating, and modifying goals. The performance loop in our model is probably the clearest manifestation of a teleologically driven motor because it depends on the perceived gap in output quality. As a result of this loop, more paths are enacted to improve output quality and narrow the output quality gap.

Dialectic. The dialectic motor explains change as occurring through the synthesis of oppositional forces and has been featured in some theories of routine dynamics (e.g., Salvato & Rerup, 2018; Turner & Rindova, 2012). In ProjectBQ, we can see a dialectical interaction between mechanisms that reinforce complexity (i.e., performance loop, revision loop, delay loop) and mechanisms that assert a balancing pressure on complexity (i.e., reinforcement loop, cut-back loop, motivation loop).

Evolution. Evolution has been an influential metaphor in routine dynamics (Feldman & Pentland, 2003), but it is difficult to operationalize in empirical research. The problem in applying this motor to real situations is the unit of analysis: What entity is being varied, selected, and retained in the population? In a computer-based simulation, Pentland et al. (2012) used whole performances of the routine (whole paths) as the phenotypes being selected. However, the participants in ProjectBQ enacted their paths one step at a time. The evolutionary model breaks down because there is no clear-cut phenotype that is being selected based on its fitness. It does not make sense to argue that there were multiple, competing patterns of action subject to evolutionary pressure.

Patterning. When the entity undergoing change is an organizational routine (or a group of routines), patterning provides a more natural explanation of how change happens. As situated actions unfold in the performance of the work, patterns are (re)enacted. Each step enacts what the routine is becoming (Tsoukas & Chia, 2002). Patterning provides a processual description for change that is grounded in the performance of the routine. By performing their work, the participants in ProjectBQ were patterning their work as well. The causal loops that we identified in ProjectBQ exemplify the kinds of factors that influence the tendency for the network to change or remain the same by adding or dropping paths.

Future Work: Routine Dynamics as Network Dynamics

By locating actions in a network of possible paths, the path-based perspective sets the stage for a rich new set of possibilities for organizational research. One particularly promising way forward builds on the idea we introduced above: routines dynamics as network dynamics.

We suggest two complementary mechanisms that are consistent with patterning: repetitive reinforcement (Sutton & Barto, 2018) and morphogenesis (Archer, 2010).

Repetitive reinforcement. When reinforced, paths in a narrative network become the ruts in the road that make the pattern repetitive and recognizable. They are more likely to be followed in future repetitions. In ProjectBQ, we have evidence that repetition leads to the reinforcement of specific edges in the graph. Repetitive reinforcement is visible, even with a small amount of data in a highly variable creative project context.

Repetitive reinforcement is a well-established mechanism for learning by doing (Levitt & March, 1988; March, 1991) and routine formation (Cohen & Bacdayan, 1994). Repetition of a known pathway exemplifies what March (1991) termed “exploitation.” Repetition provides an occasion for refinement and learning. To the extent that repetitive reinforcement is the dominant mode of patterning, it should tend to promote lock-in and inertia (Schulz, 2008).

Morphogenesis. The morphogenetic perspective of structure and action “unravel[s] the dialectical interplay between structure and action” in sequential cycles of structural conditioning, social interaction, or structural elaboration (Archer, 2010: 228). Morphogenesis provides a countervailing mechanism for patterning that can lead to structural elaboration (i.e., change). Archer (2010: 247) theorized that degrees of freedom and stringency of constraints are preconditions for understanding morphogenesis: “the specification of degrees of freedom and stringency of constraints makes it possible to theorize about variations in voluntarism and determinism (and their consequences).” In Archer’s (2010) theory of morphogenesis, conditions with high degrees of freedom and low stringency of constraints are hypothesized to favor change.

The narrative network framework provides a starting point for operationalizing the conditions for morphogenesis. By *degrees of freedom*, Archer (2010) meant the set of choices available for actors to

do something new or different in a given situation, such as forming a new path. At any point in the narrative network, we can operationalize degrees of freedom as the out-degree of the current node (the number of outwardly directed edges). This number will vary throughout the network because each node will have a specific out-degree. Transportation examples, such as getting to work via walking, bus, bike, or train, provide a good way to illustrate degrees of freedom. In principle, one could extend the degrees of freedom to include other, as yet unformed, paths (e.g., riding a motorcycle or hailing a ride service).

By *stringency of constraints*, Archer (2010) referred to the fact that not all degrees of freedom are equally free to every actor at every time and place. Situational factors naturally make some options more expensive, difficult, or risky. For example, bad weather might impose a constraint that makes cycling difficult or impossible; the need to carry a lot of equipment might require a taxi or even a truck. In principle, stringency of constraints could be modeled as a cost function that could be assigned to each edge in the graph, depending on context. This information is over and above what would normally be included in a narrative network.

We can interpret morphogenesis in terms of more familiar concepts, such as exploration (March, 1991) and innovation (Garud, Tuertscher, & de Ven, 2013). The innovation process has been increasingly conceptualized as an ongoing accomplishment without a well-defined end point, and not simply a journey with predefined stages (Garud Gehman, Kumaraswamy & Tuertscher, 2016, 2013; Obstfeld, 2012; Van de Ven, Polley, Garud, & Venkataraman, 1999). Creating and maintaining possibilities is important for these kinds of exploratory, emergent activities. Future work could investigate how exploration and innovation are encouraged by conditions or efforts to sustain the variety of pathways.

Practical Implications: Managing Paths

The practical value of understanding paths is well established in business process management (Dumas, La Rosa, Mendling, & Reijers, 2018). Commercial software products such as Celonis (<https://www.celonis.com>) and Fluxicon Disco (<http://fluxicon.com/>) have been inspired by the idea that you cannot manage what you cannot see. Processes are difficult to see, so these products use digital trace data to make work processes visible, to monitor process execution, and to provide feedback on

process conformance and other aspects of process performance. They are primarily intended for use on highly structured, computerized processes where digital data are readily available.

However, business process management tends to emphasize conformance and compliance—sticking to prescribed paths. While avoiding deviant paths is important, one can also think of paths in terms of opportunities. For example, the entrepreneurship literature has talked about exploiting entrepreneurial opportunities through an effectual approach (Sarasvathy, 2001, 2009) that leverages contingencies, rather than a planned approach based on assumptions that reduce uncertainty. Under conditions of uncertainty, it might be more beneficial to encourage path creation rather than conformance, as the former would provide insights into the potential opportunities created (Alvarez & Barney, 2007; Garud, Kumaraswamy, & Karnøe, 2010).

The path-based perspective may allow us to consider managerial decision in a way that takes morphogenesis into consideration. When we ask, “What conditions will lead to favorable outcomes?” we can consider the process that connects the antecedents and the consequences (Abbott, 1990). Rather than reducing the intervening process to one best path or one best practice, the path-based perspective encourages us to consider the space of possibilities. Thus, instead of studying antecedents and consequences to identify the best path (and trying to follow it), we might consider the antecedents and consequences of expanding or contracting the space of possibilities. Such an approach might be useful in managing other types of complex emergent phenomena, such as innovation (Dougherty, 2016; Dougherty & Dunne, 2011; Garud et al., 2016) and dynamic team processes (Cronin, Weingart, & Todorova, 2011; Kozlowski & Chao, 2018; Srikanth, Harvey, & Peterson, 2016).

Limitations

A limitation of a path-based perspective pertains to the availability of data to construct the narrative network. In this study, the agile software development methodology created a useful archival record: the scrum sheets. In other settings, it may be difficult to gather data that provides a meaningful trace of actions over time. Nevertheless, recent advances in technologies for sensing and capturing fine-grained behaviors offer promise in overcoming this limitation (Kozlowski, Chao, Chang, & Fernandez, 2015; Lazer et al., 2009).

The temporal unit of analysis (the time window) matters. Because ProjectBQ was enacted in a series of sprints, our view of the dynamics of the project is based on a compilation of snapshots. The flow we see is more like a storyboard than a finished movie. It is likely that changing the timeframe, or the “exposure time,” of the picture could generate a different sense of flow. This is particularly true if a process is changing quickly. In the case presented here, three-week sprints made a natural division; in other settings, the appropriate timeframe would be a matter of researcher judgment.

In recreating the task sequences in our data, we had to assume that tasks were performed sequentially. However, some tasks were performed concurrently, but we were not able to capture such relationships with current methods. To the extent that concurrent activities are interdependent, our method is likely to understate complexity. Nevertheless, the general trajectory of complexity was consistent with the project narrative, which was based on other data sources. Both of these limitations—temporal granularity and concurrency—point to the importance of having multiple sources of data to contextualize and interpret processual phenomena, as we have done here.

Our methodology for estimating the number of paths in the narrative network has a number of limitations as well. First, it assumes that the directed graph is a complete, accurate representation of the underlying process. In practice, processes are difficult to observe—they change constantly and there is good reason to expect that any particular enactment is ephemeral, at best. Additionally, it may not be clear where a process starts or where it finishes, and researchers will have to rely on their judgment for such decisions. However, the estimator used to compute the number of paths (see Appendix B) addresses this concern because there is no need to specify the start or end for the process.

Second, our metric for complexity does not account for differences in difficulties in performing within-role and cross-role handoffs. In the case of ProjectBQ, for example, moving from Experimentation to Building was likely to be more challenging when the handoff was with someone from the same function (e.g., from one Artist to another) compared to someone from another function (e.g., from an Artist to the Software Engineer). In addition, because our method of estimating paths is based on the structure of the narrative network, it could overstate or understate the actual number of paths in some

situations. These issues can be explored in future research.

CONCLUSION

Organizational phenomena are widely acknowledged to be complex and dynamic. However, there are few studies in organizational and management research that have explicitly accounted for the dynamics of complexity. Our research demonstrates how the concept of paths allows us to see the dynamic *patterning* (Danner-Schröder & Geiger, 2016; Feldman, 2016a; Turner & Rindova, 2018) of routine actions. We show how paths change over time, and develop a theory about the mechanisms driving these dynamics. The concept of paths as a “new way of seeing” can be applied to different processual phenomena with emergent characteristics to advance organizational and managerial science by developing new theories about the dynamics of these phenomena.

REFERENCES

- Abbott, A. 1990. A primer on sequence methods. *Organization Science*, 1: 375–392.
- Adler, P. S., & Obstfeld, D. 2007. The role of affect in creative projects and exploratory search. *Industrial and Corporate Change*, 16: 19–50.
- Alvarez, S. A., & Barney, J. B. 2007. Discovery and creation: Alternative theories of entrepreneurial action. *Strategic Entrepreneurship Journal*, 1: 11–26.
- Archer, M. S. 2010. Morphogenesis versus structuration: On combining structure and action. *British Journal of Sociology*, 61: 225–252.
- Aroles, J., & McLean, C. 2016. Rethinking stability and change in the study of organizational routines: Difference and repetition in a newspaper-printing factory. *Organization Science*, 27: 535–550.
- Barabási, A., & Albert, R. 1999. Emergence of scaling in random networks. *Science*, 286: 509–512.
- Barley, S. R. 1986. Technology as an occasion for structuring: Evidence from observations of CT scanners and the social order of radiology departments. *Administrative Science Quarterly*, 31: 78–108.
- Berente, N., Lyytinen, K., Yoo, Y., & King, J. L. 2016. Routines as shock absorbers during organizational transformation: Integration, control, and NASA's enterprise information system. *Organization Science*, 27: 551–572.
- Bertels, S., Howard-Grenville, J., & Pek, S. 2016. Cultural molding, shielding, and shoring at Oilco: The role of culture in the integration of routines. *Organization Science*, 27: 573–593.
- Bucher, S., & Langley, A. 2016. The interplay of reflective and experimental spaces in interrupting and reorienting routine dynamics. *Organization Science*, 27: 594–613.
- Butts, C. T. 2008. A relational event framework for social action. *Sociological Methodology*, 38: 155–200.
- Chemero, A. 2003. An outline of a theory of affordances. *Ecological Psychology*, 15: 181–195.
- Chia, R. 2016. Process, practices, and organizational competitiveness: Understanding dynamic capabilities through a process philosophical worldview. In A. Langley & H. Tsoukas (Eds.), *The SAGE handbook of process organization studies*: 593–600. Beverly Hills, CA: Sage.
- Cohen, M. D. 2007. Reading Dewey: Reflections on the study of routine. *Organization Studies*, 28: 773–786.
- Cohen, M. D., & Bacdayan, P. 1994. Organizational routines are stored as procedural memory: Evidence from a laboratory study. *Organization Science*, 5: 554–568.
- Cohendet, P. S., & Simon, L. O. 2016. Always playable: Recombining routines for creative efficiency at Ubisoft Montreal's video game studio. *Organization Science*, 27: 614–632.
- Cronin, M. A., Weingart, L. R., & Todorova, G. 2011. Dynamics in groups: Are we there yet? *Academy of Management Annals*, 5: 571–612.
- Danner-Schröder, A., & Geiger, D. 2016. Unravelling the motor of patterning work: Toward an understanding of the microlevel dynamics of standardization and flexibility. *Organization Science*, 27: 633–658.
- Davis, J. A. 1970. Clustering and hierarchy in interpersonal relations: Testing two graph theoretical models on 742 sociomatrices. *American Sociological Review*, 35: 843–851.
- Deken, F., Carlile, P. R., Berends, H., & Lauche, K. 2016. Generating novelty through interdependent routines: A process model of routine work. *Organization Science*, 27: 659–677.
- Dewey, J. 1938/2008. Logic: The theory of inquiry. In J. A. Boyston (Ed.), *The Later Works, 1925–1953*, Vol 12. Carbondale, IL: Southern Illinois Press.
- Dittrich, K., Guérard, S., & Seidl, D. 2016. Talking about routines: The role of reflective talk in routine change. *Organization Science*, 27: 678–697.
- Dougherty, D. 2016. *Taking advantage of emergence: Productively innovating in complex innovation systems*. Oxford, U.K.: Oxford University Press.
- Dougherty, D., & Dunne, D. D. 2011. Organizing ecologies of complex innovation. *Organization Science*, 22: 1214–1223.

- Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. 2018. *Fundamentals of business process management*, vol. 1 (2nd ed.). Berlin: Springer-Verlag.
- Ebert, C., & Cain, J. 2016. Cyclomatic complexity. *IEEE Software*, 33: 27–29.
- Emirbayer, M., & Mische, A. 1998. What is agency? *American Journal of Sociology*, 103: 962–1023.
- Feldman, M. S. 2000. Organizational routines as a source of continuous change. *Organization Science*, 11: 611–629.
- Feldman, M. S. 2016a. Routines as process: Past, present, and future. In J. Howard-Grenville, C. Rerup, A. Langley, & H. Tsoukas (Eds.), *Organizational routines: A process perspective*: 23–46. Oxford, U.K.: Oxford University Press.
- Feldman, M. S. 2016b. Making process visible: Alternatives to boxes and arrows. In A. Langley & H. Tsoukas (Eds.), *The Sage handbook of process organization studies*: 625–635. London, U.K.: Sage.
- Feldman, M. S., & Pentland, B. T. 2003. Reconceptualizing organizational routines as a source of flexibility and change. *Administrative Science Quarterly*, 48: 94–118.
- Feldman, M. S., Pentland, B. T., D’Adderio, L., & Lazaric, N. 2016. Beyond routines as things: Introduction to the special issue on routine dynamics. *Organization Science*, 27: 505–513.
- Freeman, L. C. 1977. A set of measures of centrality based on betweenness. *Sociometry*, 40(1): 35–41.
- Garud, R., et al. 2016. From the process of innovation to innovation as process. In A. Langley & H. Tsoukas (Eds.), *The SAGE handbook of process organization studies*: 451–466. Thousand Oaks, CA: SAGE.
- Garud, R., Kumaraswamy, A., & Karnøe, P. 2010. Path dependence or path creation? *Journal of Management Studies*, 47: 760–774.
- Garud, R., Tuertscher, P., & de Ven, A. H. V. 2013. Perspectives on innovation processes. *Academy of Management Annals*, 7: 775–819.
- Gaskin, J. E., Berente, N., Lyytinen, K., & Yoo, Y. 2014. Toward generalizable sociomaterial inquiry: A computational approach for zooming in and out of sociomaterial routines. *Management Information Systems Quarterly*, 38: 849–871.
- Glaser, B. G., & Strauss, A. L. 1967. *The discovery of grounded theory: Strategies for qualitative research*. Chicago, IL: Aldine de Gruyter.
- Hærem, T., Pentland, B. T., & Miller, K. D. 2015. Task complexity: Extending a core concept. *Academy of Management Review*, 40: 446–460.
- Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., & Morris, M. 2008. statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, 24: 1–11.
- Hernes, T. 2014. *A process theory of organization*. Oxford, U.K.: Oxford University Press.
- Holland, P. W., & Leinhardt, S. 1977. A dynamic model for social networks. *Journal of Mathematical Sociology*, 5: 5–20.
- Howard-Grenville, J. A., & Rerup, C. 2017. A process perspective on organizational routines. In A. Langley & H. Tsoukas (Eds.), *The SAGE handbook of process organization studies*: 323–337. London, U.K.: Sage.
- James, W. 1909/1996. *A pluralistic universe: Hibbert lectures at Manchester College on the present situation in philosophy*. Lincoln, Nebraska: University of Nebraska Press.
- Jarzabkowski, P., Lê, J., & Spee, P. 2016. Taking a strong process approach to analyzing qualitative process data. In A. Langley & H. Tsoukas (Eds.), *The SAGE handbook of process organization studies*: 237–253. Thousand Oaks, CA: SAGE.
- Kozlowski, S. W., & Chao, G. T. 2018. Unpacking team process dynamics and emergent phenomena: Challenges, conceptual advances, and innovative methods. *American Psychologist*, 73: 576–592.
- Kozlowski, S. W., Chao, G. T., Chang, C. H., & Fernandez, R. 2015. Team dynamics: Using “Big Data” to advance the science of team effectiveness. In S. Tonidandel, E. King, & J. M. Cortina (Eds.), *Big data at work: The data science revolution and organizational psychology*: 273–309. New York, NY: Routledge Academic.
- Langley, A. 1999. Strategies for theorizing from process data. *Academy of Management Review*, 24: 691–710.
- Langley, A., Smallman, C., Tsoukas, H., & Van de Ven, A. H. 2013. Process studies of change in organization and management: Unveiling temporality, activity, and flow. *Academy of Management Journal*, 56: 1–13.
- Langley, A., & Tsoukas, H. 2016a. Introduction: Process thinking, process theorizing and process researching. In A. Langley & H. Tsoukas (Eds.), *The SAGE handbook of process organization studies*. Thousand Oaks, CA: Sage.
- Langley, A., & Tsoukas, H. 2016b. *The SAGE handbook of process organization studies*. Thousand Oaks, CA: Sage.
- Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Christakis N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., Van Alstyne, M. 2009. Life in the network: the coming age of computational social science. *Computational Social Science*, 323: 721–723.
- Leenders, R. T. A., Contractor, N. S., & DeChurch, L. A. 2016. Once upon a time: Understanding team processes as relational event networks. *Organizational Psychology Review*, 6: 92–115.

- Levitt, B., & March, J. G. 1988. Organizational learning. *Annual Review of Sociology*, 14: 319–338.
- March, J. G. 1991. Exploration and exploitation in organizational learning. *Organization Science*, 2: 71–87.
- McCabe, T. J. 1976. A complexity measure. *IEEE Transactions on Software Engineering*, 2: 308–320.
- McPherson, M., Smith-Lovin, L., & Cook, J. M. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27: 415–444.
- Mead, G.H. 1934/1962. *Mind, self and society*. Chicago, IL: University of Chicago Press.
- Mesle, C., & Dibben, M. R. 2016. Whitehead's process relational philosophy. *The SAGE handbook of process organization studies*, 29–42. Thousand Oaks, CA: SAGE.
- Moe, N. B., Dingsøyr, T., & Dybå, T. 2010. A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, 52: 480–491.
- Moody, J., McFarland, D., & Bender-deMoll, S. 2005. Dynamic network visualization. *American Journal of Sociology*, 110: 1206–1241.
- Nicolini, D. 2009. Zooming in and out: Studying practices by switching theoretical lenses and trailing connections. *Organization Studies*, 30: 1391–1418.
- Obstfeld, D. 2012. Creative projects: A less routine approach toward getting new things done. *Organization Science*, 23: 1571–1592.
- Pentland, B. T. 1992. Organizing moves in software support hot lines. *Administrative Science Quarterly*, 37: 527–548.
- Pentland, B. T. 1999. Building process theory with narrative: From description to explanation. *Academy of Management Review*, 24: 711–724.
- Pentland, B. T., & Feldman, M. S. 2007. Narrative networks: Patterns of technology and organization. *Organization Science*, 18: 781–795.
- Pentland, B. T., Feldman, M. S., Becker, M. C., & Liu, P. 2012. Dynamics of organizational routines: A generative model. *Journal of Management Studies*, 49: 1484–1508.
- Pentland, B. T., Hærem, T., & Hillison, D. 2010. Comparing organizational routines as recurrent patterns of action. *Organization Studies*, 31: 917–940.
- Pentland, B. T., & Liu, P. 2017. Network models of organizational routines: Tracing associations between actions. In S. Jain & R. Mir (Eds.), *Routledge companion to qualitative research in organization studies*. New York, NY: Routledge.
- Pentland, B. T., Liu, P., Kremser, W., & Hærem, T. Forthcoming. Dynamics of drift in digitized processes. *Management Information Systems Quarterly*. Published online ahead of print.
- Pentland, B. T., Recker, J., & Wyner, G. 2015. A thermometer for interdependence: Exploring patterns of interdependence using networks of affordances. Paper presented at the Thirty sixth international conference on information systems, Fort Worth, TX.
- Pentland, B. T., Recker, J., & Wyner, G. 2016. Conceptualizing and measuring interdependence between organizational routines. Paper presented at the Thirty seventh international conference on information systems, Dublin, Ireland.
- Pentland, B. T., Recker, J., & Wyner, G. 2017. Rediscovering handoffs. *Academy of Management Discoveries*, 3: 284–301.
- Rerup, C., & Feldman, M. S. 2011. Routines as a source of change in organizational schemata: The role of trial-and-error learning. *Academy of Management Journal*, 54: 577–610.
- Rescher, N. 1996. *Process metaphysics: An introduction to process philosophy*. Albany, NY: Suny Press.
- Rudolph, J. W., Morrison, J. B., & Carroll, J. S. 2009. The dynamics of action-oriented problem solving: Linking interpretation and choice. *Academy of Management Review*, 34: 733–756.
- Salvato, C., & Rerup, C. 2018. Routine regulation: Balancing conflicting goals in organizational routines. *Administrative Science Quarterly*, 63: 170–209.
- Sarasvathy, S. D. 2001. Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. *Academy of Management Review*, 26: 243–263.
- Sarasvathy, S. D. 2009. *Effectuation: Elements of entrepreneurial expertise*, London, U.K.: Edward Elgar Publishing.
- Schulz, M. 2008. Staying on track: A voyage to the internal mechanisms of routine reproduction. In M. Becker (Ed.), *Handbook of organizational routines*: 228–257. Cheltenham, U.K.: Edward Elgar.
- Shotter, J. 2006. Understanding process from within: An argument for “withness”-thinking. *Organization Studies*, 27: 585–604.
- Snijders, T. A. 2001. The statistical evaluation of social network dynamics. *Sociological Methodology*, 31: 361–395.
- Snijders, T. A. B., van de Bunt, G. G., & Steglich, C. E. G. 2010. Introduction to stochastic actor-based models for network dynamics. *Social Networks*, 32: 44–60.
- Song, M., Günther, C. W., & Van der Aalst, W. M. 2008. Trace clustering in process mining. In D. Ardagna, M. Mecella, & J. Yang (Eds.), *Business Process Management Workshops*, Vol 17: 109–120. Berlin, Heidelberg: Springer.
- Srikanth, K., Harvey, S., & Peterson, R. 2016. A dynamic perspective on diverse teams: Moving from the dual-

- process model to a dynamic coordination-based model of diverse team performance. *Academy of Management Annals*, 10: 453–493.
- Strauss, A. L. 1993. *Continual permutations of action*. New York, NY: Aldine de Gruyter.
- Strike, V. M., & Rerup, C. 2016. Mediated sensemaking. *Academy of Management Journal*, 59: 880–905.
- Suchman, L. A. 1987. *Plans and situated actions: The problem of human-machine communication*. New York, NY: Cambridge University Press.
- Sutherland, J., & Sutherland, J. J. 2014. *Scrum: The art of doing twice the work in half the time*. New York, NY: Crown Business.
- Sutton, R. S., & Barto, A. G. 2018. *Reinforcement learning: An introduction*. Cambridge, MA: MIT press.
- Tiwari, U., & Kumar, S. 2014. Cyclomatic complexity metric for component based software. *ACM SIGSOFT Software Engineering Notes*, 39: 1–6.
- Tsoukas, H., & Chia, R. 2002. On organizational becoming: Rethinking organizational change. *Organization Science*, 13: 567–582.
- Turner, S. F., & Rindova, V. 2012. A balancing act: How organizations pursue consistency in routine functioning in the face of ongoing change. *Organization Science*, 23: 24–46.
- Turner, S. F., & Rindova, V. P. 2018. Watching the clock: Action timing, patterning, and routine performance. *Academy of Management Journal*, 61: 1253–1280.
- Van de Ven, A. H., Polley, D., Garud, R., & Venkataraman, S. 1999. *The innovation journey*. New York, NY: Oxford University Press.
- Van de Ven, A. H., & Poole, M. S. 1995. Explaining development and change in organizations. *Academy of Management Review*, 20: 510–540.
- Volkoff, O., & Strong, D. M. 2017. Affordance theory and how to use it in IS research. In R. D. Galliers & M. K. Stein (Eds.), *The Routledge companion to management information systems*: 232–245. New York, NY: Routledge.
- Wasserman, S., & Faust, K. 1994. *Social network analysis: Methods and applications*. New York, NY: Cambridge University Press.
- Weick, K. E. 1979. Cognitive processes in organizations. *Research in Organizational Behavior*, 1: 41–74.
- West, D. B. 2001. *Introduction to graph theory*, vol. 2. Upper Saddle River, NJ: Prentice Hall.
- Whitehead, A. N. 1929/1978. *Process and reality: An essay in cosmology*. New York, NY: Free Press.
- Wood, R. E. 1986. Task complexity: Definition of the construct. *Organizational Behavior and Human Decision Processes*, 37: 60–82.



Kenneth T. Goh (kennethgoh@smu.edu.sg) is an assistant professor of strategic management at the Lee Kong Chian School of Business, Singapore Management University. He received his PhD from The Tepper School of Business, Carnegie Mellon University. His research explores routines and processes in innovation and entrepreneurship.

Brian T. Pentland (pentland@broad.msu.edu) is the Main Street Capital Partners Endowed Professor in the Broad College of Business at Michigan State University. His work has appeared in *Academy of Management Review*, *Administrative Science Quarterly*, *Journal of Management Studies*, *Management Science*, *MIS Quarterly*, *Organization Science*, *Organization Studies*, and *ReverbNation*, as well as on YouTube and elsewhere.



APPENDIX A

COUNTING SIMPLE PATHS IN A DIRECTED GRAPH

This appendix explains a method for counting simple paths in a directed graph, and contains code for a MatLab function that implements this method.

Overview of Algorithm

The algorithm is a breadth-first search that finds all of the simple paths (sequences of connected nodes) from start to finish. The algorithm follows all edges leading out from the start node. Each connected node indicates a partial path. Then, for each partial path, it follows all of the edges leading out from the last node of the path. Each partial path is stored. New paths are added only if they are unique. If a path revisits any node, it is removed from the list of stored paths. Thus, the algorithm only counts *simple paths* (West, 2001)—paths that do not include any node more than once. When a path reaches the finish node, it is added to the list of completed paths. The algorithm continues until the graph has been exhaustively searched. It is a “brute force” enumeration of all simple paths.

MatLab Function for Counting Paths

This function requires three inputs: (1) an adjacency matrix for the directed graph that describes the task or process, (2) the source (starting point) of the task, and (3) the sink (stopping point) of the task. The function produces two outputs: (1) the number of simple paths from start to finish, and (2) a list of the simple paths from start to finish.

```
function [ simple_paths, list_of_unique_paths ] = ...
    task_complexity_index(AM, source, sink)

% This code counts the number of simple paths (no
% cycles) in a
% directed graph. Paths start at the source and end
% at the sink.

% INPUTS:
% AM: adjacency matrix for the directed graph that
% represent the task
% source: starting point for task
% sink: ending point for task

% OUTPUTS:
% simple_paths = number of simple paths
% list_of_unique_paths = cell array of strings that
% describe the paths

% Data structures for keep track of unique paths
```

```
paths_completed = containers.Map();
paths_in_progress = containers.Map();

% Use CAPITAL N for size of adjacency matrix
N = size(AM, 1);

% Convert adjMtx to 0/1 only
AM = (AM > 0);

% Take out the diagonal, since self-connected
vertices do not add paths
AM(eye(N)==1) = 0;

%%%%%%%%%%
% The main loop traverses the graph until there are
no more
% simple paths to find.
% Finished paths are stored in data.completed_paths

% start with the source to initialize the paths_in_
progress
s = add_to_paths(source,next_nodes(AM, source));

% then loop until done
loop_count = 0;
simple_paths = 0;
while paths_in_progress.Count ≥ 1

% the function “path_search” loops through all
of the paths in
% progress to see if they can be completed or
continued.
% The status flag is not used
status_flag = path_search(AM);

% the loop count is just used to display progress
if so desired
loop_count = loop_count+1;

% Uncomment these statements to view the process
% disp(strcat('Depth=', num2str(loop_count), ...
% ' PathsInProgress:', num2str(paths_in_
progress.Count),...
% ' PathsCompleted:', num2str(paths_
completed.Count));
% show_paths(paths_in_progress)
% show_paths(paths_completed)

% If you want to set a ceiling, you can do it here.
if paths_completed.Count ≥ 1000000
% re-initialize the list of paths in progress to
stop the search
paths_in_progress = containers.Map();
disp('*** over 1,000,000 paths found. Limit
ing count. ***');
end
end % paths_in_progress loop
```

```

% assign the total and the list
simple_paths = paths_completed.Count;
list_of_unique_paths = keys(paths_completed);

% disp(strcat('Total simple paths = ',num2str(
simple_paths)));

return;

% *****
*****

% loop through paths in progress and extend
them until completed
function ss_status = path_search(adjMtx)

for p_in_prog = keys(paths_in_progress)
pnp = str2num(p_in_prog{1});
ss_status = add_to_paths(pnp, next_
nodes(adjMtx,pnp));
end

end

% look at the end of the current path to get the
next nodes
function nlist = next_nodes(adjMtx, current_
path)

% use that node to get the list of next nodes
nlist = find(adjMtx(current_path(end,:),) >0);

% check if any are already on the path
nlist = setdiff(nlist,current_path);

end

% Put paths in paths_completed or paths_in_
progress
% Keep going until all possible paths are found.
function sstatus = add_to_paths(path_in, next_
node_list)

path_in_key = mat2str(path_in);

% stop if there is nowhere to go
if numel(next_node_list) == 0
if isKey(paths_in_progress, path_in_key)
remove(paths_in_progress, path_in_key);
end
sstatus = 0;
return;
else
sstatus = 1;
end

% loop through all of the potential next nodes
for next_node = next_node_list

% make sure the node has not been visited on
this path
if ~any(path_in == next_node)

```

```

% append the next node and store it
path_out = [path_in, next_node];
path_out_key = mat2str(path_out);

% if the path is done, then save it in completed set
if path_out(end) == sink
paths_completed(path_out_key) = path_out;
sstatus = 1;
else
if numel(path_out) ≥ 3*N % path is too long...
if isKey(paths_in_progress, path_out_key)
remove(paths_in_progress, path_out_key);
end
sstatus = 0;
else
paths_in_progress(path_out_key) = 1; % dummy
end
end
if isKey(paths_in_progress, path_in_key)
remove(paths_in_progress, path_in_key);
end
end
end
end
end
% this function is only used for debugging
function show_paths(c)
for v = keys(c)
v
end
end

end % of main function

```

APPENDIX B

FUNCTION FOR ESTIMATING PATHS IN DIRECTED GRAPH

The brute-force counting method in Appendix A provides a reference against which we can assess methods for estimating task complexity. However, counting paths in a network is known to be a “#P-complete” problem (Bax, 1994): the number of paths cannot be counted in polynomial time. As a practical matter, as the size and density of the network increases, no amount of computing resources can solve the problem. The number of paths can be enumerated for smaller networks (Rubin, 1978; Bax, 1994), but for larger networks it must be estimated (Roberts & Kroese, 2007).

To develop an alternative that is computationally tractable, we build on the method introduced by McCabe (1976) for estimating the complexity of a software module. This measure, called cyclomatic complexity, is still in use as a measure of software complexity (Ebert & Cain, 2016; Tiwari & Kumar, 2014). McCabe (1976) represented the execution paths in a block of code as a directed graph, and then used the

number of nodes and edges to estimate the number of execution pathways. McCabe also adjusted for sub-routines, which act like nodes and tend to reduce the number of execution paths:

Complexity ~ Edges–Nodes–Subroutines

The analogy to task complexity is straightforward. As in Hærem et al. (2015), the nodes in the graph are the “required acts” in a task and the edges represent the connections between those acts. Note that this functional form contradicts the widely held intuition that a greater number of required acts increases complexity (Wood, 1986). The interpretation here is subtler: for a given number of nodes, it is the number of edges that drives complexity.

We fit this function to the results of the exact algorithm in Appendix A using a set of simulated data ($n = 73,200$). The simulated data included 100 random repetitions for each size of network from $10 \leq nodes \leq 100$, with varying levels of network density. The networks were simulated so that there was always at least one valid path. Empirically, we found that the best fit involved a logarithmic transformation of the dependent variable, as theorized by Hærem et al. (2015). We also found that it made no difference if we counted the number of paths or the sum of the number of edges along all of the paths. The results are shown in Table B1. Regression diagnostics are shown in Figure B1, which shows standardized residuals and the

TABLE B1
Fitting the Estimate to the Exact Model

Descriptive Statistics ($n = 73,200$)			
	Min.	Avg	Max.
Nodes	10	54	100
Edges	9	67	138
Paths	1	120	25,838

TABLE B2
Regression Results Predicting Number of Paths in Simulated Networks

DV = Log_{10} (simple paths)	
Nodes	-0.079*** (.000)
Edges	0.080*** (.000)
Const.	0.120** (.003)
R^2	0.889
F	291,982.5***

*** $p < 0.001$

cumulative probability (P-P) plot. Over a wide range of conditions, the simple estimate of network paths correlates with the exact count quite well ($r = 0.94$).

Finally, we wanted to ensure that our estimate is accurate when a graph has a single path from source to sink. When there is a single path, there is one edge between each pair of nodes, so edges = nodes - 1. Therefore, when there is a single path, $\text{log}_{10}(1) = 0$. Adjusting the model to fit this analytical boundary condition results in this formula for computing task complexity based on a directed graph that represents the task:

$$\begin{aligned} \text{Enacted task complexity} &= \text{log}_{10}(\text{simple paths}) \\ &= .08 * (\text{edges} - \text{nodes} + 1) \end{aligned}$$

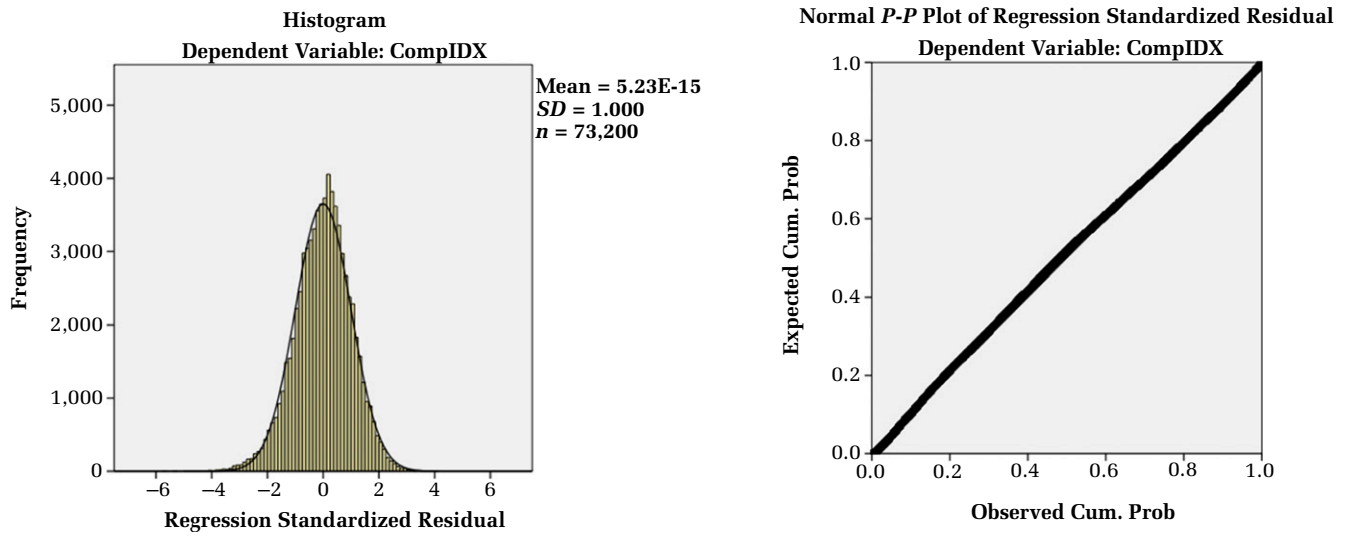
REFERENCES

Bax, E. T. 1994. Algorithms to count paths and cycles. *Information Processing Letters*, 52: 249–252.

Roberts, B., & Kroese, D. P. 2007. Estimating the number of s-t paths in a graph. *Journal of Graph Algorithms and Applications*, 11: 195–214.

Rubin, F. 1978. Enumerating all simple paths in a graph. *IEEE Transactions on Circuits and Systems*, 25: 641–642.

FIGURE B1
Diagnostic Results



Notes: P-P plot refers to a probability-probability plot of two cumulative distribution functions to compare how closely two data sets agree.

Copyright of Academy of Management Journal is the property of Academy of Management and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.