

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection Lee Kong Chian School Of
Business

Lee Kong Chian School of Business

6-2019

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 PENTLAND

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. From actions to paths to patterning: Toward a dynamic theory of patterning in routines. (2019). *Academy of Management Journal*. Research Collection Lee Kong Chian School Of Business.

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 libIR@smu.edu.sg.

1
2
3
4
5 **From Actions to Paths to Patterning:**
6 **Towards a Dynamic Theory of Patterning in Routines**
7
8
9
10

11 **Kenneth T. Goh**

12 Singapore Management University
13 50 Stamford Road, #05-01
14 Singapore 178899
15 kennethgoh@smu.edu.sg
16
17

18 **Brian T. Pentland**

19 Michigan State University
20 Business College Complex
21 632 Bogue St. N270
22 East Lansing, MI 48824
23 pentland@bus.msu.edu
24
25
26
27
28
29
30
31
32

33 Published in Academy of Management Journal, 2019 August, advance online.

34 <https://doi.org/10.5465/amj.2018.0042>

35 Submitted version

36 Creative Commons Attribution-Noncommercial-No Derivative Works 4.0 License
37
38
39
40
41
42
43
44
45

46 **Acknowledgements.** We gratefully acknowledge the thoughtful feedback from Deputy Editor Pratima
47 Bansal, three exceptional reviewers, and participants in the 2017 Academy of Management Journal New
48 Ways of Seeing paper development workshop at the Ivey Business School, the 2018 Asian Management
49 Research Consortium, the 2018 Academy of Management Big Data and Managing in a Digital Economy
50 Special Conference, the 2018 Interdisciplinary Network for Group Research, and the Singapore
51 Management University Strategy and Organisation group brown bag seminar series. We thank Hashmat
52 Habibzadah, Daniela Chang, and Gabriela Dedelli for invaluable research assistance and acknowledge
53 with deep gratitude the time our informants at GameSG dedicated to this project. We thank Carnegie
54 Mellon University, Ivey Business School, Singapore Management University, and Michigan State
55 University for financial support.
56
57
58
59
60

1
2
3
4
5
6
7

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

8

ABSTRACT

9
10
11
12
13
14
15
16
17
18
19
20
21

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.

22
23
24

Keywords: Routine dynamics, narrative network, network paths, task complexity, process research

25
26

INTRODUCTION

27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

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

45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

In this paper, we introduce the concept of paths as a way for 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).

1
2
3 We apply this lens in the context of a new product development project, specifically video game
4 development. When paths are repetitive and recognizable, they represent performances of a
5 routine (Feldman & Pentland, 2003; Obstfeld, 2012). In the project we studied, some paths were
6 repetitive and recognizable, but there was also constant change, making it a good context to
7 study routine dynamics. We use evidence from this project to theorize about a central problem in
8 routine dynamics: what drives a pattern of action to become more or less varied?
9

10
11 We build on the concept of patterning (Danner-Schröder & Geiger, 2016; Feldman,
12 2016a; Turner & Rindova, 2018) as a way to describe routine dynamics. We conceptualize
13 patterning as the formation of new paths and the dissolution of old paths in the narrative network
14 that describes the routine (Pentland & Feldman, 2007). The general approach is analogous to
15 established models of social network dynamics (Snijders, 2001; Snijders, van de Bunt, &
16 Steglich, 2010), but instead of examining ties between a fixed set of *actors*, we trace paths
17 between a constantly changing set of *actions*. We found that action patterns change dramatically
18 over time depending on project needs and explain generic mechanisms that lead to more (or
19 fewer) paths being enacted in the narrative network. While these mechanisms relate to Van de
20 Ven and Poole's (1995) classic typology of change motors, we propose that patterning can be
21 seen as a novel motor of change for routine dynamics.
22
23

24
25 A path-based focus is not a minor methodological twist. It goes hand in glove with a
26 theoretical perspective called "strong" process theory (Hernes, 2014; Langley & Tsoukas, 2016a;
27 Tsoukas & Chia, 2002). Strong process theory offers a radical, process-centric ontology of the
28 social world. Tracing the formation and dissolution of paths over time provides a concrete way
29 to operationalize strong process theory in empirical research on routine dynamics. Our path-
30 based approach offers a new way of seeing and measuring how the patterns of action in a routine
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 have changed. This allows us to describe and theorize about the mechanisms that drive routine
4
5 dynamics.
6

7 8 **THEORY**

9
10 The philosophical roots of strong process theory can be traced to Whitehead, James,
11 Mead, and Dewey and more recently the work of Chia (2016), Hernes (2014), Rescher (1996),
12 Shotter (2006) and others (see Langley & Tsoukas, 2016b). The basic insight is simple. As
13 Weick (1979: 95) observed, “organizations are grounded in interlocked behaviors rather than
14 interlocked people.” Putting actions in the foreground, rather than actors, aligns with the view
15 that the social world is a continually unfolding process (Strauss, 1993; Tsoukas & Chia, 2002).
16 Thus, the “dynamic, unfolding process becomes the primary unit of analysis rather than the
17 constituent elements themselves” (Emirbayer & Mische, 1998: 287). This strong process view is
18 widely adopted in research on routine dynamics (Howard-Grenville & Rerup, 2017). In the
19 following sections, we review the routine dynamics literature and explain how paths can provide
20 a new way of seeing and characterizing the dynamics of organizational routines.
21
22
23
24
25
26
27
28
29
30
31
32
33

34 **Routine dynamics**

35
36 Routine dynamics focuses on the stability and change of organizational routines from a
37 processual perspective (Feldman et al., 2016). Routines are repetitive, but because each
38 performance of a routine unfolds over time, it can always unfold in a new direction (Feldman &
39 Pentland, 2003). While there are many factors that help routines “stay on track” (Schulz, 2008),
40 routines are not constrained to follow pre-defined paths (Feldman, 2000; Feldman & Pentland,
41 2003; Feldman et al., 2016). Future paths are influenced by past paths, but not determined by
42 them. Furthermore, as Feldman et al. (2016) point out, some routines are not very routine: they
43 embody an enormous number of possible paths (Hærem, Pentland, & Miller, 2015; Pentland,
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 Hærem, & Hillison, 2010). All routines exhibit what (Cohen, 2007) called “pattern-in-variety,”
4
5 but some are more varied than others and at the same time, the patterns may be changing.
6
7

8 This points to a central puzzle in routine dynamics: what makes a pattern of action more
9
10 or less varied? In routine dynamics, variety enables change (Feldman, 2016a; Pentland, Liu,
11
12 Kremser, & Hærem, In press). A pattern of action that is more varied encompasses more paths,
13
14 with more possibilities for divergence/change. A pattern of action that is less varied encompasses
15
16 fewer paths, with fewer possibilities for divergence/change. However, the theoretical problem of
17
18 what drives patterning is not explained: Why do patterns of action stay the same or change over
19
20 time? There is also the methodological problem of seeing and quantifying pattern-in-variety
21
22 (Cohen, 2007). We cannot research this phenomenon if we cannot see it.
23
24
25

26 The growing body of field research on routine dynamics has focused on explanations of
27
28 stability and change. It elaborates on the concept of endogenous change as theorized by Feldman
29
30 and Pentland (2003) and points to the importance of exogenous factors as well. For example, in
31
32 their study of compliance routines in oil exploration, Bertels and colleagues (2016) show how
33
34 routines can be shielded from and shored up against external interventions. The routines remain
35
36 stable, although this stability requires effort and continual maintenance. In contrast, in their
37
38 study of NASA’s implementation of an enterprise information system, Berente and colleagues
39
40 (2016) show that routines can change through unanticipated local adaptation. As Barley (1986)
41
42 observed when CT scanners were introduced into radiology departments, Berente et al. (2016)
43
44 found that new technology can lead to new patterns of interaction in a workplace. At the
45
46 organizational level, Rerup and Feldman (2011) demonstrate how organizational routines
47
48 coevolve with organizational schema through different types of “trials” and “errors.”
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 The formation of new routines has also been a topic of considerable interest. For
4
5 example, in the context of video game development, Cohendet and Simon (2016) describe a
6
7 process of forming new routines through deliberately breaking, partitioning, and recombining
8
9 aspects from different routines in response to an organizational disruption that required them to
10
11 shift from efficiency to make room for creativity. Deken et al (2016) showed how flexing,
12
13 stretching, and inventing generated novel actions and outcomes in an automotive supplier that
14
15 was developing a new line of information-based services. Meetings (Aroles & McLean, 2016)
16
17 and spaces (Bucher & Langley, 2016) provide opportunities for questioning, reflection, and
18
19 thought experiments as participants work out new routines (Dittrich, Guérard, & Seidl, 2016).
20
21
22
23

24 This fieldwork provides evidence that routines do, in fact, change over time in systematic
25
26 ways and has led to a refined understanding of factors driving stability and change in routines.
27
28 Feldman et al. (2016) note that these field studies have generally employed situated actions as
29
30 the unit of *observation* and patterns of action as the unit of *analysis*. Ironically, the unit of
31
32 analysis (the pattern of interdependent action that makes up the routine) has been less visible.
33
34 The literature on routine dynamics theorizes about patterns of action, often without measuring or
35
36 visualizing those patterns (Feldman, 2016b).
37
38
39

40 Feldman (2016a) suggests that one way forward is to focus on the “inseparability or
41
42 mutual constitution of actions and patterning” (p. 38). Patterning exemplifies the process of
43
44 dynamic unfolding described by Emirbayer and Mische (1998), because routines are performed
45
46 one step at a time. Step by step, situated actions enact recognizable paths. Paths represent a
47
48 missing link between situated actions and repetitive patterns. By tracing paths, we can begin to
49
50 connect actions and patterns.
51
52
53
54
55
56
57
58
59
60

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., In press). 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 and 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) describe 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 and colleagues (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.

 INSERT TABLE 1 HERE

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 Chapter 13), 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

1
2
3 models for predicting dynamics (Snijders et al., 2010) and for visualizing dynamics (Handcock,
4 Hunter, Butts, Goodreau, & Morris, 2008; Moody, McFarland, & Bender-deMoll, 2005).

7 Relational event models (Butts, 2008; Leenders, Contractor, & DeChurch, 2016) that predict the
8 likelihood of a relational event (i.e., interpersonal action) between two parties provide a way to
9 model social network dynamics in continuous time.
10
11
12
13

14 We conceptualize narrative network dynamics in an analogous manner to social network
15 dynamics: as the formation and dissolution of network edges. However, narrative networks and
16 social networks are fundamentally different ways to see the social world. While social networks
17 represent the ties between actors, narrative networks represent sequential relations between
18 actions or events. In narrative networks, nodes are not individuals with cognition, motivation,
19 and other personal characteristics. Consequently, the mechanisms that drive the dynamics of
20 social networks do not apply. For example, it does not make sense for actions to be attracted to
21 each other and become sequentially related on the basis of that attraction. Thus, to theorize about
22 the dynamics of narrative networks, we need to start from scratch. For this purpose, we turn to
23 the concept of network paths.
24
25
26
27
28
29
30
31
32
33
34
35
36

37 **Network Paths.** In any kind of network, a *path* is defined as a sequence of connected
38 nodes (West, 2001). However, the interpretation of paths is different in different kinds of
39 networks. In a social network, a path counts the number of “hops” or degrees of separation
40 between individuals in the network. The shortest path provides a measure of distance between
41 nodes. It is an indication of connectivity between pairs of nodes and can be used to identify
42 nodes or ties that are critical for connectivity (Freeman, 1977; Wasserman & Faust, 1994: 105).
43
44
45
46
47
48
49
50

51 In a narrative network, a path represents a sequence of actions that might be used to carry
52 out part of an overall routine or process. Paths can also be considered as recipes for action or
53
54
55
56
57
58
59
60

1
2
3 stories: they describe how a process has been or could be performed. Like any recipe or story, it
4 is carried out one step at a time. The nodes in the network are the actions and the edges represent
5 the movement from one action to the next, connecting those actions into paths.
6
7
8
9

10 Thus, we see a path as a sequence of steps enacted over time. Building on Strauss (1993),
11 Obstfeld (2012) used the term *trajectory* to refer to the same basic idea. Obstfeld (2012: 1574)
12 defined a trajectory as “a sequence of interdependent actions involving multiple actors.” In
13 business process management, paths are often referred to as “traces” (Song, Günther, & Van der
14 Aalst, 2008). While we are referring to the same concept, we prefer the term *path* because it
15 emphasizes the graph theoretic interpretation (West, 2001).
16
17
18
19
20
21
22
23

24 **Steps to Paths to Patterns.** The narrative network provides a theoretical explanation of
25 how enacting different steps influences the possible paths in a routine. When we add or remove
26 steps (edges) from a narrative network, it changes the set of possible paths. It creates (or
27 removes) possible ways of getting things done. For example, if a new bus route or subway line
28 opens (or closes), it may create (or remove) a possible path for getting to work. As a result of
29 these changes, new paths become available and old paths become unavailable. Each possible
30 path contributes to the overall pattern.
31
32
33
34
35
36
37
38
39

40 In general, adding actions (nodes) and/or handoffs (edges) will tend to *increase* the
41 number of paths. Removing actions (nodes) and/or handoffs (edges) will tend to *decrease* the
42 number of paths. These relationships are not hypotheses; they are based on mathematical
43 properties of directed graphs. The question for organizational research is: What mechanisms
44 drive these dynamics?
45
46
47
48
49
50

51 METHODS

52 We conducted a field study of a video game development project team that involved
53 being “in the flow” to capture longitudinal data through observations, interviews, and archival
54
55
56
57
58
59
60

1
2
3 materials. In-depth field work provided the fine-grained detail necessary to bring the phenomena
4 to life (Feldman et al., 2016; Jarzabkowski, Lê, & Spee, 2016).
5
6

7 8 **Research Setting**

9 The setting for our study is a project team, ProjectBQ, at a video game development
10 studio, GameSG (both pseudonyms), based in a mid-Atlantic city in the United States. During
11 the period of data collection, GameSG was a 10-year old studio that employed approximately 60
12 employees, mostly under 30 years of age, with expertise in software engineering, game design,
13 and technical art. Prior development projects at GameSG included games on various platforms
14 (e.g., mobile phones, stand-alone entertainment systems, TV plug-in games, internet browser
15 games) for a wide spectrum of clients that included video game publishers, media
16 conglomerates, theme parks, and a startup toy company.
17
18
19
20
21
22
23
24
25
26

27 Project teams in GameSG were usually composed of members with expertise in one of
28 the following skill sets - game design, software engineering, technical art, script writing,
29 animation, sound composition, and project management. The composition of team members in
30 ProjectBQ was typical in this regard. The team was led by a core group of functional “leads”
31 consisting of the producer, a lead designer, a technical lead, and an art lead. Each lead was
32 responsible for coordinating work in that functional domain and acting as a gatekeeper for the
33 quality of work produced. Project leads were also directly involved on high level decisions about
34 the design and functionality of the game. The producer managed deadlines, the pace of work, and
35 access to resources for the team. They played a boundary-spanning role between the team and
36 other stakeholders such as GameSG management, other project teams, and the client. The team
37 size for ProjectBQ ranged from 8 to 15 developers over a 14-month period.
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

53 ProjectBQ was funded by a non-profit with the goal of promoting anti-drug messages
54 through unstructured learning methods. The project was a “serious” game intended to teach
55
56
57
58
59
60

1
2
3 teenagers about resisting peer pressure in high-risk situations (e.g., substance abuse, risky
4 behavior). The game was themed as a fantasy game where the hero protagonist is a mouse that is
5 attempting to protect his tribe from the corrupting influence of the villain antagonist. Players
6 progressed in the game by visiting new worlds to battle enemies. Battles were turn-based and
7 were won by whether the player picked the right move that would best counter the one chosen by
8 the computer. Although the game had “fantasy characters”, players had to make decisions based
9 on real world situations. Describes Producer1,

10
11
12
13
14
15
16
17
18
19 “It is not direct messages saying, “Don’t do drugs.” What it’s saying is, “Here are some
20 situations that you’re not going to be comfortable with in real life. Here are responses and
21 ways in which you can handle those situations without feeling like a nerd or an outcast,
22 or like you’re going to lose your friends or things like that.” (Producer1)

23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
ProjectBQ was typical of the game development projects at GameSG in that the stages of
development followed a standard sequence of a pre-production, production, and refinement. The
pre-production 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 co-designer, 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

1
2
3 eventually built and delivered to the client, albeit behind schedule. Despite the negativity in the
4 development process, the game was found to have moderate success in improving adolescent
5 players' ability to identify pressuring situations as well as recognize and practice healthy
6 responses. The game was also a finalist for several gaming awards and was rated 4.2 stars on
7 Google Play and 4 stars on iTunes, out of a possible 5 stars.
8
9

10
11
12 We picked video game development as an exemplary setting for studying routine
13 dynamics because it is a collective task that is ambiguous and emergent: there are an endless
14 number of possibilities for combining elements to create a game. Video games are an interactive
15 virtual experience produced by a computer program onto a display device that people engage in
16 for entertainment. Although games are also used in more "serious" settings such as education
17 and training simulations, there is always an element of interactivity and engagement with the
18 player. However, how this interactivity and engagement manifests in the context of the game is
19 rarely obvious at the outset of game development (Cohendet & Simon, 2016).
20
21
22
23
24
25
26
27
28
29
30
31
32

33 These characteristics of video game development can be considered a type of creative
34 project (Obstfeld, 2012). Creative projects consist of an emergent trajectory of interdependent
35 action initiated and orchestrated by multiple actors to introduce change into a social context. The
36 nature of these departures could be in the form of new elements, or new linkages between
37 familiar elements. The ambiguous means and ends of creative projects imply that "repetition is
38 not a guide on what to do next" (Obstfeld, 2012: 1571) as the trajectory of action required to
39 create the video game does not follow a set plan. On a continuum of routine and non-routine
40 actions, ProjectBQ is clearly at the non-routine end of the continuum (Adler & Obstfeld, 2007;
41 Obstfeld, 2012).
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

Data Collection

Our research design incorporated data from archival materials, non-participant observation, and interviews. Data was 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

1
2
3 actions were then grouped by stories and sequenced according to the following order: 1) when
4 the task ended, 2) when it was started, 3) the order in which the action was added to the database.
5
6 The last criterion was necessary to determine the ordering of actions that shared similar start
7
8 dates and end dates.
9
10

11
12 Between May 2011 to August 2012, the first author was a non-participant observer on
13 ProjectBQ. These observations included team meetings ($n = 39$), client meetings ($n = 7$), and
14 play test sessions ($n = 4$). Team meetings included daily fifteen-minute “scrum” meetings ($n =$
15 29) where team members met to schedule and coordinate their tasks for the day, retrospective
16 meetings where they reviewed work processes ($n = 2$), and general discussions about the project
17 ($n = 8$). During these meetings, notes were taken about the purpose of the meeting, what was said
18 and by whom, and the author’s impressions of what transpired during the meeting.
19
20

21
22 In addition to data from observations, both ad hoc informal ($n = 11$) and formal semi-
23 structured interviews ($n = 4$) were conducted with team members. The informal interviews
24 focused on getting status updates on the project while formal semi-structured interviews were
25 about 60 minutes long and focused on gaining an in-depth understanding of specific episodes
26 during the project. Interviews were conducted with the producer, the two tech leads, the art lead,
27 a designer, and a software engineer. Archival materials such as project schedules, planning
28 documents, meeting notes, and budgets were also accessed and referenced to establish rich
29 insights into the events surrounding the actions taken by the team.
30
31

32 **Data Analysis**

33
34 In keeping with our goal of seeing and theorizing about patterning as it was enacted over
35 the course of the project, we analyzed data chronologically as a narrative. The data analysis
36 consisted of three main steps: (1) constructing a series of narrative networks that represent
37
38
39
40
41
42
43
44
45
46

1
2
3 patterns of action throughout the project; (2) computing the properties of each network; and (3)
4
5 constructing a project narrative to interpret and theorize about the dynamics of those patterns.
6

7
8 **Constructing narrative networks.** Narrative networks were constructed in the following
9
10 steps: 1) code the data into sequences that can be used to construct narrative networks; 2) bracket
11
12 the data into windows of analysis that correspond to project sprints; and 3) construct and
13
14 visualize the networks through a software application called *ThreadNet* (Pentland et al., 2015,
15
16 2016).
17
18

19 The first step, coding the data into sequences, required coding the final list of 2,428
20
21 activities from the scrum sheets according to the actions and roles involved in each activity. We
22
23 used a constant comparative process (Glaser & Strauss, 1967) to develop task categories with the
24
25 help of two research assistants. Categories were developed by iterating between the first author's
26
27 familiarity with the context, field notes, and other archival documents to understand the intent of
28
29 the task. This process involved forming initial clusters of tasks to minimize differences within
30
31 clusters while maximizing differences between clusters. An initial set of categories were then
32
33 developed from these clusters. New tasks were then compared with earlier tasks in the same
34
35 category. If a newly categorized task appeared to be different from other tasks in the same
36
37 category, this would be reconciled by attempting to refine the definitions and properties of these
38
39 categories to accommodate the new data. This process of constantly comparing new data with
40
41 existing codes was continued until a level of stability was reached. From twelve initial
42
43 categories, the list was ultimately reduced to the following six categories: Administration,
44
45 Experimenting, Building, Revision, Refinement, and Testing (Table 2). Figure 1 shows the
46
47 distribution of these categories over time.
48
49
50
51
52
53
54
55
56
57
58
59
60

INSERT FIGURE 1 ABOUT HERE

1
2
3
4
5
6
7
8 Roles were coded in a similar approach to coding actions. The primary actor responsible
9
10 for each task in the database was categorized into an organizational role by the first author based
11
12 on the researcher's familiarity with the research setting. These roles were Design, Art, Tech, and
13
14 Analytics (Figure 2). Together with the actions, these roles define the possible actions in the
15
16 Analytics (Figure 2). Together with the actions, these roles define the possible actions in the
17
18 narrative network. The six roles and six task categories meant that there were potentially 36
19
20 unique role-task categories. The 36 unique role-task categories were applied to the 2,428 time-
21
22 ordered events gathered from the scrum sheets to create a set of 159 coded sequences. These
23
24 coded sequences become the input for creating a series of narrative networks for the project as it
25
26 progressed.
27

28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

INSERT FIGURE 2 ABOUT HERE

33 The second step involved bracketing narrative networks into windows of analysis.
34
35 ProjectBQ was implemented using an agile software development methodology (Moe, Dingsøyr,
36
37 & Dybå, 2010; Sutherland & Sutherland, 2014), which meant that the project was divided into
38
39 three-week long phases called "Sprints." For our analysis, we bracketed the data (Langley,
40
41 1999) into three-week windows that corresponded with the dates for each Sprint.
42
43

44 The third step involved constructing and visualizing the networks through a software
45
46 application, *Threadnet* (Pentland et al., 2015, 2016). *ThreadNet* was used to convert the coded
47
48 sequences into a narrative network for each sprint. The application traces the coded sequences of
49
50 action to create networks. Each kind of coded action becomes a node in the network. Adjacent
51
52
53
54
55
56
57
58
59
60

1
2
3 pairs of coded actions become edges in the network. Although this procedure can be performed
4
5 manually, the software is faster and less error-prone.
6

7
8 ***Computing properties of narrative networks.*** Once the networks for each sprint were
9
10 constructed, we computed their properties. The two basic properties that define any network are
11
12 (a) the list of nodes and (b) the list of edges (West, 2001). For our purposes, we simply needed
13
14 to count the number of nodes and edges in each graph. These counts are provided automatically
15
16 *ThreadNet* (and other network analysis tools) and are shown in Table 3 (see below).
17
18

19 To estimate the number of paths in the network, we used a simple formula based on
20
21 McCabe's (1976) concept of cyclomatic complexity:
22

$$23 \quad (1) \quad \mathbf{Estimated\ paths} = 10^{0.08 * (Edges - Nodes + 1)}$$

24
25 Stated in English, the estimated number of paths in a network is an exponential function
26
27 of the difference between the number of edges and the number of nodes. For a given number of
28
29 nodes, increasing the number of edges will increase the estimated number of paths. The constant
30
31 (0.08) is derived empirically by fitting this equation to thousands of simulated networks with a
32
33 known number of paths. The derivation, validation, and limitations of this formula are provided
34
35 in Appendices A and B. The complexity index (Hærem et al., 2015) is computed as a logarithmic
36
37 function of the number of estimated paths.
38
39
40
41

42 ***Constructing the overall project narrative.*** We constructed a timeline of events from
43
44 interviews with informants. These interviews were professionally transcribed and analyzed using
45
46 nVivo software to identify periods, major events, and the critical actors associated with the
47
48 temporal unfolding of the project (Langley, 1999; Pentland, 1999). We drew on the first author's
49
50 observations of the project team to validate our timeline of the project. Each observational event
51
52
53
54
55
56
57
58
59
60

1
2
3 was dated and summarized. We then compared the events provided by informants with these
4
5 observations to validate the timeline.
6

7
8 To create a more detailed narrative, we augmented the basic project timeline by iterating
9
10 between the interviews and the observations with an emphasis on the contextual circumstances
11
12 surrounding interpretations of why events occurred, individual thoughts and feelings in response
13
14 to actors and incidents, and histories. This narrative provided a depth of insight into the
15
16 unfolding project that extended temporally across the past and into the future, and across actors
17
18 that included individuals, the team, and external stakeholders.
19
20

21 **FINDINGS**

22
23 We report our findings in two parts. In the first part, we describe *four phases of*
24
25 *patterning in ProjectBQ*. During each phase, the number of paths in the narrative network
26
27 increased or decreased dramatically. In the second part, we combine our qualitative data about
28
29 the project with quantitative metrics about the narrative network to theorize about the
30
31 *mechanisms that drive routine dynamics*.
32
33

34 **Four phases of patterning in ProjectBQ**

35
36 As a creative project, ProjectBQ involved a lot of change. Figure 3 shows how the
37
38 complexity of the project (indexed by the number of paths) changed over time. Table 3 shows
39
40 the narrative network for each sprint, plus the number of distinct nodes/edges and the number of
41
42 paths added/removed from one sprint to the next. Table 3 also mentions the mechanisms that
43
44 drive dynamics, which are explained in the next section. Here, we discuss the project in four
45
46 phases of patterning that correspond to distinct changes within the project.
47
48
49

50
51 -----
52 INSERT FIGURE 3 ABOUT HERE
53 -----
54
55
56
57
58
59
60

INSERT TABLE 3 ABOUT HERE

1
2
3
4
5
6
7
8 **Phase one: Sprints 1 to 4 (Increasing Complexity).** In phase one, complexity increased
9
10 between Sprints 1 to 4 (adding over 43,000 paths to the network). This increase was driven by
11
12 both an increase in the number of distinct actions (from 9 to 17) as well as the number of distinct
13
14 handoffs (from 19 to 87).
15

16
17 This increase can be explained by the fact that the first phase of the project consisted of
18
19 concepting, prototyping, and developing the core mechanics of the game. Sprint 1 was
20
21 designated as the phase to develop “Initial Concepts”. Sprint 2 was initially designated as a “Pre-
22
23 Production” phase, the goal of which was for developers to rehearse the steps for producing
24
25 game assets and incorporating these assets into the game to get a sense of the production
26
27 schedule. Going through this process helps them to “make sure a lot of these later milestones
28
29 were laid out and could be accomplished” (Art1). However, Sprint 2 was later renamed as a
30
31 “Production” phase. Sprint 3 was assigned to be the phase for developing “Battle prototype”
32
33 which was a core mechanic of the game. This was to be followed by a phase for developing the
34
35 combat system and the game environment in Sprint 4, which was labelled the “Combat, Burrow”
36
37 phase. To develop the “combat” feature of the game, the Designer needed to account for
38
39 technical and aesthetic concerns which required closer collaboration, coordination, and iteration
40
41 with Tech and Art. This interdependence between developers from different functions is evident
42
43 from the doubling in distinct handoffs from Sprint 2 to Sprint 4.
44
45
46
47
48

49 **Phase two: Sprint 5 (Decreasing Complexity).** In Sprint 5, complexity decreased to
50
51 1.52. Interestingly, the number of distinct actions remain the same at 17. The decrease in
52
53 complexity is driven by the decrease in handoffs ($n = 47$), which resulted in a decrease of over
54
55 43,000 possible paths. Complexity declined in Sprint 5 because a prototype was to be delivered
56
57
58
59
60

1
2
3 to the client at the end of the sprint. As a result, most of the actions were building-related as
4
5 evident from the increase in frequency of Building tasks in Sprint 5 (Figure 2).
6

7
8 **Phase three: Sprints 6 to 8 (Surge in Complexity).** In phase three, complexity
9
10 increased between Sprints 6 to 8 (from 3.92 to 5.76). The number of distinct actions increased
11
12 slightly from Sprint 5 but remained constant throughout this phase ($n = 21$). However, the
13
14 number of distinct handoffs more than doubled from Sprint 5 (from 83 to 106). This seemingly
15
16 minor increase in handoffs led to a dramatic addition of over 500,000 possible paths. This
17
18 change is especially striking because the number of distinct actions was constant during this
19
20 phase.
21
22

23
24 This enormous increase in possible paths resulted from the ProjectBQ developers
25
26 working towards a “gamma build” deliverable that was due in Sprint 11. As the “feature lock”
27
28 deadline was in Sprint 9, there was a flurry of activity that included both Experimentation- and
29
30 Building-related actions to confirm the final features of the game in Sprint 8. We found that the
31
32 increase in handoffs was due to team members iterating between downstream roles (e.g.,
33
34 “Sound”) and tasks (e.g., “Refinement”) and upstream roles (e.g., “Design”) and tasks (e.g.,
35
36 “Experimenting”).
37
38

39
40 **Phase four: Sprints 9 to 11 (Decline in complexity).** In phase four, complexity
41
42 decreased between Sprints 9 to 11 (from 3.12 to 1.74). This decrease in complexity was caused
43
44 by both a decrease in distinct actions (from 18 to 10) and distinct handoffs (from 66 to 38). The
45
46 number of possible paths dropped off by over 500,000, mostly in sprint 9.
47
48

49
50 In phase four, there was a decline in both distinct actions and handoffs because the
51
52 feature lock deadline in Sprint 9 meant that no more changes to the design could be made.
53
54 Hence, the remaining actions were mostly Building-related. There were no longer major design
55
56
57
58
59
60

1
2
3 changes that required developers to iterate between experimenting and building, or between
4
5 functions.
6

7 **Mechanisms that drive routine dynamics**

8
9 In the second part of our findings, we draw on our findings to identify mechanisms that
10 drive the complexity of routine dynamics through the addition (or removal) of actions and
11 handoffs from the network. These mechanisms operate to varying degrees throughout the
12 performance of the project. To identify these mechanisms, we draw on causal loops
13 diagramming methods that are commonly used in system dynamics research to articulate process
14 theories (e.g., Rudolph et al., 2009; Strike & Rerup, 2016) to unpack how events unfolded in
15 ProjectBQ. We identify a total of six mechanisms that directly affect the complexity of routine
16 dynamics: reinforcement loop, performance loop, revision loop, delay loop, cut-back loop, and
17 motivation loop.
18
19
20
21
22
23
24
25
26
27
28
29

30 ***Reinforcement loop.*** Reinforcement through repetition is one of the basic mechanisms
31 of stability in routines (Cohen & Bacdayan, 1994; Schulz, 2008). In ProjectBQ, we found that
32 the frequency of a handoff in one sprint was positively related to the tendency for that handoff to
33 occur in the next sprint. If a handoff appeared more than once in a given sprint, there was a 55%
34 chance it appeared in the next sprint. If a handoff appeared more than five times in a given sprint,
35 the chance of it appearing in the next sprint increased to 95%. These findings thus provide
36 evidence of stable, repetitive patterns of action even within the context of a creative project. By
37 itself, repetition tends to reduce complexity because, in a routine with many thousands of
38 possible paths, stronger paths get reinforced and weaker paths are forgotten (Pentland et al, in
39 press). Conversely, greater complexity (more paths) reduces the chances that a particular path
40 will be repeated. We label this relationship between repetitive patterns of action and complexity
41 as the reinforcement loop (Figure 4a).
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

INSERT FIGURE 4a ABOUT HERE

1
2
3
4
5
6
7
8 **Performance loop.** Figure 4b shows the set of relationships in the performance loop that
9
10 drive actions and handoffs. Output quality gap – the gap between the client’s requirements and
11
12 the overall quality of the project team’s output – instigates the developers to act to narrow this
13
14 gap. This mechanism was evident throughout the project but manifested in different ways at
15
16 different sprints.
17

18
19 At the beginning of the project, particularly in Sprints 1 and 2, actions were taken to help
20
21 designers to understand the game mechanics and make decisions about the features and
22
23 functionalities that the game will have. Producer1 explained the process at this stage as follows:
24

25
26 “[The designer] felt that we should have a pre-production phase. Your designer needs a
27
28 pre-production process because they need to figure out what the game is and then they
29
30 need to start designing it before the tech people come in and start building it. You can’t
31
32 build something that hasn’t been figured out yet.” (Producer1)
33

34
35 The goal of the pre-production phase was for developers to rehearse the steps for
36
37 producing game assets and incorporating these assets into the game to get a sense of the
38
39 production schedule. Going through this process helps them to “make sure a lot of these later
40
41 milestones were laid out and could be accomplished” (Art1). A large proportion of actions in
42
43 Sprints 1 and 2 thus consisted largely of Experimentation actions performed by the core group of
44
45 eight developers.
46
47

48
49 In Sprint 3, the team’s headcount increased because the Tech lead lobbied senior
50
51 management to bring on more developers to the team sooner. This decision was made due to
52
53 concerns that the project had been over scoped, which would hurt their ability to meet project
54
55 deadlines. With the increase in headcount, there was also a corresponding increase in the total
56
57
58
59
60

1
2
3 frequency of actions from 65 in Sprint 2 to 98 in Sprint 3 as developers “ramp up” and move into
4 the Production phase to build the game, even while continuing to experiment with different
5 ideas. The increase in actions and handoffs enabled the team to work towards their first major
6 project milestone - to deliver a playable prototype to the client at the end of Sprint 5.
7
8
9
10
11

12 We have thus far explained how the performance loop increases actions and handoffs.
13 However, this mechanism could also reduce actions and handoffs when expectations for output
14 quality were low, such as in Sprint 5. In Sprint 5, the complexity index decreased from 4.64 in
15 Sprint 4 to 1.56. This decrease in complexity index was due to fewer handoffs since the
16 frequency of actions were approximately similar in both Sprints 4 (n = 221) and 5 (n = 224). The
17 reason there were fewer handoffs in Sprint 5 is because the developers were focused on
18 completing the prototype at the end of Sprint 5. After they had iterated on a design that they
19 thought was good enough to meet the client’s expectations for this milestone, the team then
20 focused on Building and Refining actions to build the prototype. Thus, the proportion of
21 Building actions increased from 36.7% in Sprint 4 to 67.4% in Sprint 5. Of note were the
22 decreases in Testing actions from 19.0% in Sprint 4 to 4.9% in Sprint 5, and decreases in
23 Experimenting actions from 26.7% in Sprint 4 to 14.3% in Sprint 5. Furthermore, since quality
24 expectations for prototypes were lower, developers did not need to iterate between functions and
25 revise their work as frequently, which explains the fewer handoffs and paths. This relationship
26 between output quality gap, actions and pathways, and output quality partially explains the
27 dynamics of complexity, which increased between Sprints 1 to 4 then plunged in Sprint 5.
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

49 In the second half of the project, from Sprint 6 onwards, the ProjectBQ team has a new
50 milestone to deliver a “gamma build” of the game in Sprint 11. New actions and handoffs were
51
52
53
54
55
56
57
58
59
60

1
2
3 undertaken to develop the game to meet this milestone, which led to the increase in complexity
4
5 index between Sprints 6 to 8.
6

7
8 After the feature lock deadline in Sprint 9, the game could no longer be improved by
9
10 adding or modifying game features. Consequently, this narrowed the output quality gap by
11
12 reducing quality expectations to refining or “polishing” the features that were already in place.
13
14 The narrowing of the output gap due to the feature lock deadline instigated a corresponding shift
15
16 towards Building and Refining actions, with fewer pathways across roles which led to the
17
18 decline in complexity from Sprints 9 to 11.
19
20

21
22 -----
23 INSERT FIGURE 4b ABOUT HERE
24
25 -----

26 **Revision loop.** The Performance loop also intersected with other mechanisms, one of
27
28 which is the Revision loop. As more components of the game become completed, developers
29
30 could playtest the game and learn about which features of the game to change, add, or remove.
31
32 This feedback triggers revisions to the design which led to more actions and handoffs between
33
34 roles to accomplish, creating a positive feedback cycle that we label the “Revision loop” (Figure
35
36 4c).
37
38

39
40 The Revision loop was evident in Sprints 3 and 4 of ProjectBQ where complexity
41
42 increased from 2.68 to 4.70. By Sprint 3, the team had completed an early prototype (the “Gold
43
44 Spike”) and had gone through the process of incorporating some graphical assets into the game.
45
46 Going through this production process made them aware of constraints they could not predict
47
48 before. As Artist1 explained,
49

50
51 “As we got more work done, we realized that this design wasn’t working or this spec
52
53 needed to change, which forced a rewrite of tech. It happened a lot with UI (user
54
55
56
57
58
59
60

1
2
3 interface) and it happened a lot with some of the other core mechanics, like the burrow
4 and combat.” (Artist1)
5
6

7 For example, they discovered that animated movements were too “jerky”. The team
8 narrowed down their options to either reducing the size of art assets or redesigning the combat
9 system. While reducing the size of graphics was much quicker than redesigning the game, it
10 would also reduce its quality. To figure this out, the team first experimented with reducing
11 graphical quality but later realized that they would have to change the combat system from “3 vs.
12 3” to “1 vs 1”. Thus, exploring these options involved several iterations between Art, Tech, and
13 Design, which was reflected in the high number of cross-functional handoffs in the “Combat”
14 story in Sprints 3 and 4. The iterative process also led to handoffs between actions at different
15 stages of development. In Sprints 3 and 4, some parts of the “Combat” story were in the early
16 stages of experimenting, while others were in the later stages of testing and revision. An example
17 of an experimenting task that Design was assigned to in the Combat story was “Influences for
18 Combat” (Sprint 4, Thread 22, ID 56); while an example of a later stage testing task for Tech
19 was “2nd pass on enemy AI” (Sprint 4, Thread 22, ID 59).
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

38 -----
39 INSERT FIGURE 4c ABOUT HERE
40 -----
41

42 **Delay loop.** The Delay loop is a positive feedback loop that indirectly affects actions and
43 handoffs through revisions. In ProjectBQ, the frequent revisions to game design and features
44 slowed down the project team’s progress. Developers thus had less time to implement features
45 that they had originally planned for, resulting in further revisions that increase actions and
46 handoffs.
47
48
49
50
51
52

53 In revising the combat system in Sprint 3 for example, “changes to [the] core mechanic
54 mess[ed] up productivity” as the requirements for many related features “changed drastically”
55
56
57
58
59
60

1
2
3 resulting in “rework[ing] some stuff [they] had done before” or “rewrit[ing] code from scratch”
4
5 (Artist1). Revisions thus throw the production schedule for the entire project off track – not only
6
7 do they have less time to accomplish their remaining tasks, but there is also more work due to the
8
9 revisions.
10

11
12 Another example of revisions causing delays was evident between Sprints 6 to 8.
13
14 ProjectBQ was characterized by frequent revisions where “there was a new idea or new situation
15
16 that will then change” (Artist 1) roughly every two weeks. Consistent with this claim, we found
17
18 evidence of an iterative process in these sprints from the presence of upstream roles (e.g.,
19
20 Design) and actions (e.g., Experimenting) performed together with further downstream roles
21
22 (e.g. Sound) and actions (e.g., Refinement) during these sprints. By then, the team was already
23
24 behind schedule. Sprint 6 was intended to be the phase where they developed the social elements
25
26 of the game and was labeled “Global quest, friends list, bring friends on missions, analytics,
27
28 tutorials”. However, the list of tasks was still dominated by those for “Combat”, “Missions”, and
29
30 “Burrow”, which were goals for Sprints 4 and 5.
31
32
33

34
35 Not only did frequent revisions cause delays by slowing down the completion of goals,
36
37 but they also caused delays because the frequent revisions led developers to intentionally leave
38
39 their tasks uncompleted in anticipation of further revisions. As described by Artist1,
40

41
42 “The guys get to a point where Art wouldn't actually be making any final art for anything
43
44 because we weren't sure [about] spending that time. Let's say that it's going to take you
45
46 ten hours to make a final piece of art today. Well guess what? No one's ever going to get
47
48 more than five hours at any task, because we don't know what's going to get cut. If you
49
50 have twenty things you need to do, instead of spending ten hours on each of those tasks,
51
52
53
54
55
56
57
58
59
60

1
2
3 we're going to go through all of that for five hours. Hopefully, we'll have something to
4 show for you.” (Artist1)
5
6

7 The frequent revisions created the expectation that more changes were forthcoming. This
8 expectation, coupled with having “twenty things you need to do”, led to a more cautious
9 approach where tasks would only be partially completed to minimize any loss in time. While this
10 approach might have saved time for each individual, it slowed down the team’s progress further
11 because instead of completing a task on schedule, tasks were completed only when they became
12 a high priority which was usually when it was behind schedule and was urgently needed to be
13 completed. As a result of these delays, the main production phase was extended by four sprints in
14 Sprint 6, and the overall schedule was extended by two sprints which Producer1 reported to be
15 the first of multiple extensions over the course of the project.
16
17
18
19
20
21
22
23
24
25
26
27

28 Just as revisions caused delays, we found that delays also instigated more revisions. In
29 ProjectBQ, the team adapted to having less time by “chopping” certain features that cannot be
30 completed in the time remaining (Tech1). At the same time however, when a feature is
31 simplified (i.e., such as switching from 3 vs. 3 to 1 vs. 1 combat) the game becomes “not as
32 exciting” and the developers feel compelled to “respond by making other things more exciting,
33 more engaging” (Artist JD). Thus, even as the project scope was reduced, new features had to be
34 redesigned which increases actions and handoffs. Revisions and delays were therefore engaged
35 in a positive feedback cycle which we label the “Delay loop” (Figure 4d).
36
37
38
39
40
41
42
43
44
45
46

47 -----
48 INSERT FIGURE 4d ABOUT HERE
49 -----
50

51 ***Cut back loop.*** The cut back loop is a balancing feedback loop between revisions and
52 delays that indirectly reduces actions and handoffs through the output quality gap. Because of the
53 high degree of interdependence between components, revisions to one component led to delays
54
55
56
57
58
59
60

1
2
3 that spilled over to other components and eventually slowed the progress of the entire project.
4
5 For example, the Nov 10 meeting in Sprint 9 shows how Design1 is blocked by Tech on the
6
7 “power scripting” tasks. The Tech team cannot proceed because they are themselves blocked on
8
9 a number of their tasks. One of the reasons for their being blocked is that Tech3 is unable to start
10
11 work until Design decides how players will “level up”. However, since Design1 is faced with
12
13 higher priority tasks, he is not able to decide on the “level up” features yet. Overall, we see that
14
15 developers are entangled in an intricate web of interdependence such that delays in one
16
17 component has a domino effect on the overall rate of progress. These delays cumulate until a
18
19 critical point where the team runs out of time. By then, developers no longer have time to iterate
20
21 and refine their work and are just trying to complete their tasks “in a crappy way” or “chopping”
22
23 features and reducing the scope of the project (Tech1). In both cases, there is a reduction in
24
25 actions and handoffs through a reduction of the output quality gap.
26
27
28
29

30
31 These dynamics were manifested in Sprints 9 to 11. A deadline for a major milestone,
32
33 delivering “gamma version”, was due at the end of Sprint 11. To meet this deadline, an internal
34
35 “feature lock” deadline was set at the end of the second week of Sprint 9. The feature lock
36
37 deadline “froze” the build because no new features were allowed to be added after the deadline.
38
39 This deadline gave assurance to developers that there were no more major changes to the game
40
41 design, which allowed them to focus on building and refining their work. However, it also
42
43 reduced their scope for making a better game - they could not improve on core design features
44
45 like game mechanics but could only improve the quality of the game by refining features such as
46
47 fixing bugs, tidying up code, improving on lighting and textures of graphics. Thus, the negative
48
49 relationship between delays and actions created a balancing feedback loop between revisions,
50
51 delays, and the output quality gap which we call the “cut-back loop” (Figure 4e).
52
53
54
55
56
57
58
59
60

INSERT FIGURE 4e ABOUT HERE

1
2
3
4
5
6
7
8 **Motivation loop.** The final mechanism we identified from our data is the “motivation
9
10 loop”, which is a balancing feedback loop between revisions and individual motivations that
11
12 reduces actions and handoffs. We found that by Sprint 6, the frequent revisions were taking a toll
13
14 on developers’ morale. Tech2 describes “a huge penalty in both morale and productivity”. This
15
16 sentiment was reaffirmed by Tech1 in the following quote:
17
18

19 “It hurts to hear when you work on something, and then you are told that, “This is going
20
21 away. Just don’t worry about it anymore, like this is no longer part of the game.” That
22
23 happens to some extent in game development, but it can happen more here.... It was just
24
25 like a double kick in the pants where all this work you did right is just getting thrown out
26
27 of the window, and now we’re going to ask you to do it [again].” (Tech1)
28
29

30 These changes left them feeling frustrated and led to a noticeable shift in individual
31
32 motivations from wanting to make the “best game possible”, to just “get it done”. This meant
33
34 iterating less frequently to refine the game and holding back ideas that could improve the user’s
35
36 experience. As Tech2 mentioned, “you lose some quality and ideas that people could have
37
38 brought up” when they became focused on “just cranking away”. Revisions thus escalated until a
39
40 critical point where developers became frustrated and pulled back their efforts. This created a
41
42 balancing reinforcing loop between revisions, frustration, and actions which we label as the
43
44 “motivation loop” (Figure 4f).
45
46
47
48

49 Adding to this frustration amongst the developers, the delays also meant that they had to
50
51 work over time for several weeks to complete the game. Even then, the project was completed
52
53 two months behind schedule and without many of the initial features that had been initially
54
55 planned for.
56
57
58
59
60

INSERT FIGURE 4f ABOUT HERE

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 particular 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

1
2
3 been difficult to trace overall patterns of action involving multiple actors over time. This gap was
4
5 one of the original motivations for narrative networks: to enable field researchers to piece
6
7 together larger patterns from fragmented observations (Pentland & Feldman, 2007).
8
9

10 Routines start with situated actions (Suchman, 1987). In Project BQ, these are basic
11
12 steps required to carry out the work: creating, writing, testing, revising, etc. Some research
13
14 traditions zoom in to analyze the details of how particular actions inhabit and animate particular
15
16 situations. For example, research on affordances often zooms in on the detailed relations
17
18 between actors, actions and artifacts (Chemero, 2003; Volkoff & Strong, 2017).
19
20

21 In contrast, getting from actions to patterns involves zooming out in two distinct ways.
22
23 First, we zoom out from actions to paths by paying attention to the sequence of actions along the
24
25 path. In addition to asking “what happened?”, we explicitly ask “What happened *next*?” In doing
26
27 so, we locate each action in the context of an enacted path. The sequential relations between
28
29 actions provide the forward motion that gets work done, but it is important to realize that neither
30
31 participants nor observers are always able to see for themselves what happens next along a path.
32
33 In contemporary organizations, the next step may happen in another part of the world.
34
35
36

37 Next, when we zoom out from paths to patterns, we locate each action in the context of a
38
39 network of paths. The network summarizes enacted paths within a particular window of time. In
40
41 doing so, it reveals the possible paths forward from each action, which may be many or few.
42
43 The network of possible paths represents the emergent accomplishments (Feldman, 2000) and
44
45 the pattern of interdependent actions that define organizational routines (Feldman & Pentland,
46
47 2003). By zooming out from actions to paths to networks, the pattern of action becomes visible.
48
49
50
51 When we zoom out from individual actions to consider patterns, we see that the individual
52
53
54
55
56
57
58
59
60

1
2
3 actions are not independent. This is a defining characteristic of organizational routines (Feldman
4 and Pentland, 2003).
5
6

7
8 Placing actions in networks has important theoretical and methodological implications,
9
10 such as the relationship between actual paths and possible paths. At any moment along a given
11 path, there are possibilities for branching onto a different path. Possible paths are inferred based
12 on actual, observed edges in the graph. Since narrative networks are usually quite sparse, the
13 inferred possibilities constitute a tiny fraction of the paths that could be formed if the network
14 was fully connected. Since narrative networks are usually quite sparse, the
15 inferred possibilities constitute a tiny fraction of the paths that could be formed if the network
16 was fully connected.
17
18
19
20

21
22 In ProjectBQ, thousands of possible paths could be inferred from the enacted paths. To
23 express these possibilities, we aggregate actual paths into a narrative network (Pentland &
24 Feldman, 2007). Because it aggregates alternative paths, the network includes possibilities that
25 may never be actualized. These possibilities may be considered part of the latent structure of the
26 routine. Since many of these paths would be considered unusual or exceptional by the
27 participants, it would be inappropriate to equate the narrative network with the ostensive aspects
28 of the routine. The ostensive aspects of a routine embody normative, typical understandings that
29 tend to guide action (Feldman & Pentland, 2003). In contrast, the narrative network embodies
30 possibilities of paths that could be taken, as inferred from actual paths enacted.
31
32
33
34
35
36
37
38
39
40
41

42
43 In a simple routine, there may be only a few ways to do the work. Sparse networks with a
44 handful of paths are easy to comprehend, but it is difficult to appreciate how quickly the number
45 of paths can change and how large it can get. Our intuition is that more required actions adds
46 complexity. This intuition is captured in the concept of component complexity (Wood, 1986).
47 However, a larger number of actions is not the primary driver of complexity. For any given
48 number of actions, adding more edges (more paths) will increase complexity exponentially
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 (Hærem et al., 2015). This increase can only be estimated by counting the edges (pairs of
4 actions) that occur along the enacted paths. The methodological move from actions to paths to
5 networks enables an entirely new way of seeing task complexity. In ProjectBQ, we observed
6 orders-of-magnitude changes in the number of paths from one sprint to the next, *even though the*
7 *number of actions was the same*. This descriptive finding adds urgency to our research question:
8 what drives these dramatic changes?
9

17 **Patterning as a motor of routine dynamics**

19 In our study, we identify six causal loops that influence the number of possible paths in
20 the project from sprint to sprint. Many of these are specific to the world of videogame
21 development, so we are hesitant to generalize too broadly. However, if we step back from the
22 particulars of ProjectBQ, we can see that generic factors such as cost, quality and deadlines, can
23 influence action patterns. We can interpret these causal loops in terms of Van de Ven and
24 Poole's (1995) classic typology of change motors: life cycle, teleologic, dialectic, and
25 evolutionary. However, *patterning* provides a novel motor of change that is closer to the
26 phenomena we observed and a better theoretical fit for routine dynamics. To see how this is so,
27 first consider the traditional motors of change.
28
29

39 ***Life-cycle.*** The life-cycle motor depicts change as a progression through a prescribed
40 sequence of stages regulated by an underlying logic, program, or code. In ProjectBQ, we can see
41 the whole project as a life-cycle, from conception to final deliverable. Within the project, the
42 three-week sprints can be interpreted from a life-cycle perspective, as well, since each sprint has
43 a repetitive structure. Life-cycles provide an excellent explanation for routine repetition, but
44 they have not featured prominently as drivers of routine dynamics.
45
46
47
48
49
50
51

53 ***Teleology.*** The teleological motor explains change as a goal-directed process undertaken
54 by an entity that involves a cycle of setting, implementing, evaluating, and modifying goals. The
55
56
57
58
59
60

1
2
3 performance loop in our model is probably the clearest manifestation of a teleologically-driven
4 motor because it depends on the perceived gap in output quality. As a result of this loop, more
5
6 paths are enacted to improve output quality and narrow the output quality gap.
7
8

9
10 ***Dialectic.*** The dialectic motor explains change as occurring through the synthesis of
11
12 oppositional forces and has been featured in some theories of routine dynamics (e.g., Salvato &
13
14 Rerup, 2018; Turner & Rindova, 2012). In ProjectBQ, we can see a dialectical interaction
15
16 between mechanisms that reinforce complexity (i.e., performance loop, revision loop, delay
17
18 loop) and the mechanisms that assert a balancing pressure on complexity (i.e., reinforcement
19
20 loop, cut-back loop, motivation loop).
21
22

23
24 ***Evolution.*** Evolution has been an influential metaphor in routine dynamics (Feldman &
25
26 Pentland, 2003), but it is difficult to operationalize in empirical research. The problem in
27
28 applying this motor to real situations is the unit of analysis: what entity is being varied, selected
29
30 and retained in the population? In a computer-based simulation, Pentland et al (2012) used whole
31
32 performances of the routine (whole paths) as the phenotypes being selected. However, the
33
34 participants in ProjectBQ enacted their paths one step at a time. The evolutionary model breaks
35
36 down because there is no clear-cut phenotype that is being selected based on its fitness. It does
37
38 not make sense to argue that there were multiple, competing patterns of action subject to
39
40 evolutionary pressure.
41
42
43

44
45 ***Patterning.*** When the entity undergoing change is an organizational routine (or a group
46
47 of routines), patterning provides a more natural explanation of how change happens. As situated
48
49 actions unfold in the performance of the work, patterns are (re)enacted. Each step enacts what
50
51 the routine is becoming (Tsoukas & Chia, 2002). Patterning provides a processual description
52
53 for change that is grounded in the performance of the routine. By performing their work, the
54
55
56
57
58
59
60

1
2
3 participants in Project BQ were patterning their work, as well. The causal loops that we
4 identified in Project BQ exemplify the kinds of factors that influence the tendency for the
5 network to change or remain the same by adding or dropping paths.
6
7
8
9

10 **Future work: Routine dynamics as network dynamics**

11 By locating actions in a network of possible paths, the path-based perspective sets the
12 stage for a rich new set of possibilities for organizational research. One particularly promising
13 way forward builds on the idea we introduced above: routines dynamics as network dynamics.
14 We suggest two complementary mechanisms that are consistent with patterning: repetitive
15 reinforcement (Sutton & Barto, 2018) and morphogenesis (Archer, 2010).
16
17
18
19
20
21
22

23 ***Repetitive reinforcement.*** When reinforced, paths in a narrative network become the ruts
24 in the road that make the pattern repetitive and recognizable. They are more likely to be followed
25 in future repetitions. In ProjectBQ, we have evidence that repetition leads to the reinforcement of
26 specific edges in the graph. Repetitive reinforcement is visible, even with a small amount of data
27 in a highly variable creative project context.
28
29
30
31
32
33
34

35 Repetitive reinforcement is a well-established mechanism for learning by doing (Levitt &
36 March, 1988; March, 1991) and routine formation (Cohen & Bacdayan, 1994). Repetition of a
37 known pathway exemplifies what March (1991) termed “exploitation.” Repetition provides an
38 occasion for refinement and learning. To the extent that repetitive reinforcement is the dominant
39 mode of patterning, it should tend to promote lock-in and inertia (Schultz, 2008).
40
41
42
43
44
45

46 ***Morphogenesis.*** The morphogenetic perspective of structure and action “unravel[s] the
47 dialectical interplay between structure and action” (p. 228) in sequential cycles of structural
48 conditioning/social interaction/structural elaboration (Archer, 2010). Morphogenesis provides a
49 countervailing mechanism for patterning that can lead to structural elaboration (i.e., change).
50
51
52
53
54

55 Archer (2010, p. 247) theorized that degrees of freedom and stringency of constraints are
56
57
58
59
60

1
2
3 preconditions for understanding morphogenesis: “the specification of degrees of freedom and
4 stringency of constraints makes it possible to theorize about variations in voluntarism and
5
6 determinism (and their consequences)...” In Archer’s (2010) theory of morphogenesis,
7
8 conditions with high degrees of freedom and low stringency of constraints are hypothesized to
9
10 favor change.
11
12
13

14
15 The narrative network framework provides a starting point for operationalizing the
16
17 conditions for morphogenesis. By *degrees of freedom*, Archer (2010) means the set of choices
18
19 available for actors to do something new or different in a given situation, such as forming a new
20
21 path. At any point in the narrative network, we can operationalize degrees of freedom as the out-
22
23 degree of the current node (the number of outwardly directed edges). This number will vary
24
25 throughout the network because each node will have a specific out-degree. Transportation
26
27 examples, like getting to the work by walking, bus, bike or train, provide a good way to illustrate
28
29 degrees of freedom. In principle, one could extend the degrees of freedom to include other, as yet
30
31 unformed, paths (e.g., riding a motorcycle or hailing a ride service).
32
33
34

35
36 By *stringency of constraints*, Archer (2010) is referring to the fact that not all degrees of
37
38 freedom are equally free to every actor at every time and place. Situational factors naturally
39
40 make some options more expensive, difficult or risky. For example, bad weather might impose a
41
42 constraint that makes cycling difficult or impossible; the need to carry a lot of equipment might
43
44 require a taxi or even a truck. In principle, stringency of constraints could be modeled as a cost
45
46 function that could be assigned to each edge in the graph, depending on context. This
47
48 information is over and above what would normally be included in a narrative network.
49
50

51
52 We can interpret morphogenesis in terms of more familiar concepts such as exploration
53
54 (March, 1991) and innovation (Garud, Tuertscher, & Ven, 2013). The innovation process has
55
56
57
58
59
60

1
2
3 been increasingly conceptualized as an ongoing accomplishment without a well-defined end
4 point and not simply a journey with predefined stages (Garud et al., 2016, 2013; Obstfeld, 2012;
5 Van de Ven, Polley, Garud, & Venkataraman, 1999). Creating and maintaining possibilities is
6 important for these kinds of exploratory, emergent activities. Future work could investigate how
7 exploration and innovation are encouraged by conditions or efforts to sustain the variety of
8 pathways.
9

16 **Practical implications: Managing paths**

17
18 The practical value of understanding paths is well established in business process
19 management (Dumas, La Rosa, Mendling, & Reijers, 2018). Commercial software products
20 such as Celonis (<https://www.celonis.com>) and Fluxicon Disco (<http://fluxicon.com/>) have been
21 inspired by the idea that you cannot manage what you cannot see. Processes are difficult to see,
22 so these products use digital trace data to make work processes visible, to monitor process
23 execution and to provide feedback on process conformance and other aspects of process
24 performance. They are primarily intended for use on highly structured, computerized processes
25 where digital data are readily available.
26
27

28
29 However, business process management tends to emphasize conformance and
30 compliance: sticking to prescribed paths. While avoiding deviant paths is important, one can
31 also think of paths in terms of opportunities. For example, the entrepreneurship literature talks
32 about exploiting entrepreneurial opportunities through an effectual approach (Sarasvathy, 2001,
33 2009) that leverages contingencies rather than a planned approach based on assumptions that
34 reduce uncertainty. Under conditions of uncertainty, it might be more beneficial to encourage
35 path creation rather than conformance as the former would provide insights into the potential
36 opportunities created (Alvarez & Barney, 2007; Garud, Kumaraswamy, & Karnøe, 2010).
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 The path-based perspective may allow us to consider managerial decision in a way that
4 takes morphogenesis into consideration. When we ask, “what conditions will lead to favorable
5 outcomes,” we can consider the process that connects the antecedents and the consequences
6 (Abbott, 1990). Rather than reducing the intervening process to one best path or one best
7 practice, the path-based perspective encourages us to consider the space of possibilities. Thus,
8 instead of studying antecedents and consequences to identify the best path (and trying to follow
9 it), we might consider the antecedents and consequences of expanding or contracting the space of
10 possibilities. Such an approach might be useful in managing other types of complex emergent
11 phenomena such as innovation (Dougherty, 2016; Dougherty & Dunne, 2011; Garud et al., 2016)
12 and dynamic team processes (Cronin, Weingart, & Todorova, 2011; Kozlowski & Chao, 2018;
13 Srikanth, Harvey, & Peterson, 2016).

28 **Limitations**

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

43 The temporal unit of analysis (the time window) matters. Because ProjectBQ was enacted
44 in a series of sprints, our view of the dynamics of the project is based on a compilation of
45 snapshots. The flow we see is more like a story board rather than a finished movie. It is likely
46 that changing the timeframe or the “exposure time” of the picture could generate a different
47 sense of flow. This is particularly true if a process is changing quickly. In the case presented
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 here, three-week “sprints” made a natural division, but in other settings, the appropriate time
4
5 frame would be a matter of researcher judgment.
6

7
8 In recreating the task sequences in our data, we had to assume that tasks were performed
9
10 sequentially. However, some tasks were performed concurrently but we were not able to capture
11
12 such relationships with current methods. To the extent that concurrent activities are
13
14 interdependent, our method is likely to understate complexity. Nevertheless, the general
15
16 trajectory of complexity was consistent with the project narrative, which was based on other data
17
18 sources. Both of these limitations – temporal granularity and concurrency – point to the
19
20 importance of having multiple sources of data to contextualize and interpret processual
21
22 phenomena, as we have done here.
23
24
25

26 Our methodology for estimating the number of paths in the narrative network has a
27
28 number of limitations, as well. First, it assumes that the directed graph is a complete, accurate
29
30 representation of the underlying process. In practice, processes are difficult to observe - they
31
32 change constantly and there is good reason to expect that any particular enactment is ephemeral,
33
34 at best. Additionally, it may not be clear where a process starts or where it finishes, and
35
36 researchers will have to rely on their judgment for such decisions. However, the estimator used
37
38 to compute the number of paths (see Appendix B) addresses this concern because there is no
39
40 need to specify the start or end for the process.
41
42
43

44 Second, our metric for complexity does not account for differences in difficulties in
45
46 performing within-role and cross-role handoffs. In the case of ProjectBQ, for example, moving
47
48 from Experimentation to Building is likely to be more challenging when the handoff is with
49
50 someone from the same function (e.g., from one Artist to another) compared to someone from
51
52 another function (e.g., from an Artist to the Software Engineer). Also, because our method of
53
54
55
56
57
58
59
60

1
2
3 estimating paths is based on the structure of the narrative network, it could overstate or
4
5 understate the actual number of paths in some situations. These issues can be explored in future
6
7 research.
8
9

10 CONCLUSION

11 Organizational phenomena are widely acknowledged to be complex and dynamic. Yet
12
13 there are few studies in organizational and management research that explicitly account for the
14
15 dynamics of complexity. Our research demonstrates how the concept of paths allows us to see
16
17 the dynamic *patterning* (Danner-Schröder & Geiger, 2016; Feldman, 2016a; Turner & Rindova,
18
19 2018) of routine actions. We show how paths change over time and develop a theory about the
20
21 mechanisms driving these dynamics. The concept of paths as a “new way of seeing” can be
22
23 applied to different processual phenomena with emergent characteristics to advance
24
25 organizational and managerial science by developing new theories about the dynamics of these
26
27 phenomena.
28
29
30
31

32 REFERENCES

- 33
34 Abbott, A. 1990. A Primer on Sequence Methods. *Organization Science*, 1(4): 375–392.
35
36 Adler, P. S., & Obstfeld, D. 2007. The role of affect in creative projects and exploratory search.
37 *Industrial and Corporate Change*, 16(1): 19–50.
38
39 Alvarez, S. A., & Barney, J. B. 2007. Discovery and creation: Alternative theories of
40 entrepreneurial action. *Strategic Entrepreneurship Journal*, 1(1–2): 11–26.
41
42 Archer, M. S. 2010. Morphogenesis versus structuration: On combining structure and action.
43 *British Journal of Sociology*.
44
45 Aroles, J., & McLean, C. 2016. Rethinking stability and change in the study of organizational
46 routines: Difference and repetition in a newspaper-printing factory. *Organization Science*,
47 27(3): 535–550.
48
49 Barabási, A., & Albert, R. 1999. Emergence of scaling in random networks. *Science*, 286(5439):
50 509–512.
51
52 Barley, S. R. 1986. Technology as an occasion for structuring: Evidence from observations of
53 CT scanners and the social order of radiology departments. *Administrative Science
54 Quarterly*, 78–108.
55
56 Berente, N., Lyytinen, K., Yoo, Y., & King, J. L. 2016. Routines as shock absorbers during
57 organizational transformation: Integration, control, and NASA’s enterprise information
58 system. *Organization Science*, 27(3): 551–572.
59
60

- 1
2
3 Bertels, S., Howard-Grenville, J., & Pek, S. 2016. Cultural molding, shielding, and shoring at
4 Oilco: The role of culture in the integration of routines. *Organization Science*, 27(3): 573–
5 593.
- 6 Bucher, S., & Langley, A. 2016. The interplay of reflective and experimental spaces in
7 interrupting and reorienting routine dynamics. *Organization Science*, 27(3): 594–613.
- 8 Butts, C. T. 2008. 4. A Relational Event Framework for Social Action. *Sociological*
9 *Methodology*, 38(1): 155–200.
- 10 Chemero, A. 2003. An outline of a theory of affordances. *Ecological Psychology*, 15(2): 181–
11 195.
- 12 Chia, R. 2016. Process, practices, and organizational competitiveness: Understanding dynamic
13 capabilities through a process philosophical worldview. In A. Langley & H. Tsoukas (Eds.),
14 *The SAGE handbook of process organization studies*: 593–600. Beverly Hills, CA: Sage.
- 15 Cohen, M. D. 2007. Reading Dewey: Reflections on the study of routine. *Organization Studies*,
16 28(5): 773–786.
- 17 Cohen, M. D., & Bacdayan, P. 1994. Organizational routines are stored as procedural memory:
18 Evidence from a laboratory study. *Organization Science*, 5(4): 554–568.
- 19 Cohendet, P. S., & Simon, L. O. 2016. Always playable: Recombining routines for creative
20 efficiency at Ubisoft Montreal’s video game studio. *Organization Science*, 27(3): 614–632.
- 21 Cronin, M. A., Weingart, L. R., & Todorova, G. 2011. Dynamics in groups: Are we there yet?
22 *The Academy of Management Annals*, 5(1): 571–612.
- 23 Davis, J. A. 1970. Clustering and Hierarchy in Interpersonal Relations: Testing Two Graph
24 Theoretical Models on 742 Sociomatrices. *American Sociological Review*, 35(5): 843–851.
- 25 Danner-Schröder, A., & Geiger, D. 2016. Unravelling the Motor of Patterning Work: Toward an
26 Understanding of the Microlevel Dynamics of Standardization and Flexibility. *Organization*
27 *Science*, 27(3): 633–658
- 28 Deken, F., Carlile, P. R., Berends, H., & Lauche, K. 2016. Generating novelty through
29 interdependent routines: A process model of routine work. *Organization Science*, 27(3):
30 659–677.
- 31 Dittrich, K., Guérard, S., & Seidl, D. 2016. Talking about routines: The role of reflective talk in
32 routine change. *Organization Science*, 27(3): 678–697.
- 33 Dougherty, D. 2016. *Taking advantage of emergence: Productively innovating in complex*
34 *innovation systems*. Oxford University Press.
- 35 Dougherty, D., & Dunne, D. D. 2011. Organizing ecologies of complex innovation.
36 *Organization Science*, 22(5): 1214–1223.
- 37 Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. 2018. *Fundamentals of business*
38 *process management* (2nd ed.), vol. 1. Berlin: Springer-Verlag.
- 39 Emirbayer, M., & Mische, A. 1998. What is agency? *American Journal of Sociology*, 103(4):
40 962–1023.
- 41 Feldman, M. S. 2000. Organizational routines as a source of continuous change. *Organization*
42 *Science*, 11(6): 611–629.
- 43 Feldman, M. S. 2016a. Routines as process: Past, present, and future. In J. Howard-Grenville, C.
44 Rerup, A. Langley, & H. Tsoukas (Eds.), *Organizational Routines: A Process Perspective.*:
45 23–46. Oxford: Oxford University Press.
- 46 Feldman, M. S. 2016b. Making process visible: Alternatives to boxes and arrows. In A. Langley
47 & H. Tsoukas (Eds.), *The Sage handbook of process organization studies*: 625–635.
48 London: Sage.
- 49
50
51
52
53
54
55
56
57
58
59
60

- 1
2
3 Feldman, M. S., & Pentland, B. T. 2003. Reconceptualizing organizational routines as a source
4 of flexibility and change. *Administrative Science Quarterly*, 48(1): 94–118.
- 5 Feldman, M. S., Pentland, B. T., D’Adderio, L., & Lazaric, N. 2016. Beyond Routines as Things:
6 Introduction to the Special Issue on Routine Dynamics. *Organization Science*, 27(3): 505–
7 513.
- 8
9 Freeman, L. C. 1977. A set of measures of centrality based on betweenness. *Sociometry*, 35–41.
- 10 Garud, R., Gehman, J., Kumaraswamy, A., Tuertscher, P., Langley, A., et al. 2016. From the
11 process of innovation to innovation as process. *The SAGE handbook of process
12 organization studies*: 451–466.
- 13 Garud, R., Kumaraswamy, A., & Karnøe, P. 2010. Path Dependence or Path Creation? *Journal
14 of Management Studies*, 47(4): 760–774.
- 15 Garud, R., Tuertscher, P., & Ven, A. H. V. de. 2013. Perspectives on Innovation Processes.
16 *Academy of Management Annals*, 7(1): 775–819.
- 17 Gaskin, J. E., Berente, N., Lyytinen, K., & Yoo, Y. 2014. Toward Generalizable Sociomaterial
18 Inquiry: A Computational Approach for Zooming In and Out of Sociomaterial Routines. *Mis
19 Quarterly*, 38(3): 849–871.
- 20
21 Glaser, B. G., & Strauss, A. L. 1967. *The discovery of grounded theory: Strategies for
22 qualitative research*. Chicago, IL: Aldine de Gruyter.
- 23 Hærem, T., Pentland, B. T., & Miller, K. D. 2015. Task complexity: Extending a core concept.
24 *Academy of Management Review*, 40(3): 446–460.
- 25 Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., & Morris, M. 2008. statnet:
26 Software tools for the representation, visualization, analysis and simulation of network data.
27 *Journal of Statistical Software*, 24(1): 1548.
- 28
29 Hernes, T. 2014. *A Process Theory of Organization*. Oxford: Oxford University Press.
- 30 Holland, P. W., & Leinhardt, S. 1977. A Dynamic Model for Social Networks. *Journal of
31 Mathematical Sociology*, 5(1): 5–20.
- 32
33 Howard-Grenville, J. A., & Rerup, C. 2017. A process perspective on organizational routines. In
34 A. Langley & H. Tsoukas (Eds.), *The SAGE Handbook of Process Organization Studies*:
35 323–337. London: Sage.
- 36 Jarzabkowski, P., Lê, J., & Spee, P. 2016. Taking a strong process approach to analyzing
37 qualitative process data. *The SAGE Handbook of Process Organization Studies*, 237.
- 38 Kozlowski, S. W., & Chao, G. T. 2018. Unpacking team process dynamics and emergent
39 phenomena: Challenges, conceptual advances, and innovative methods. *American
40 Psychologist*, 73(4): 576.
- 41
42 Kozlowski, S. W., Chao, G. T., Chang, C. H., & Fernandez, R. 2015. Team dynamics: Using
43 “Big Data” to advance the science of team effectiveness. In S. Tonidandel, E. King, & J. M.
44 Cortina (Eds.), *Big data at work: The data science revolution and organizational
45 psychology*: 273–309.
- 46
47 Langley, A. 1999. Strategies for theorizing from process data. *Academy of Management
48 Review*, 24(4): 691–710.
- 49 Langley, A., Smallman, C., Tsoukas, H., & Van de Ven, A. H. 2013. Process studies of change
50 in organization and management: Unveiling temporality, activity, and flow. *Academy of
51 Management Journal*, 56(1): 1–13.
- 52
53 Langley, A., & Tsoukas, H. 2016a. Introduction: Process thinking, process theorizing and
54 process researching. In A. Langley & H. Tsoukas (Eds.), *The SAGE Handbook of Process
55 Organization Studies*. Thousand Oaks, CA: Sage.
- 56
57
58
59
60

- 1
2
3 Langley, A., & Tsoukas, H. 2016b. *The SAGE Handbook of Process Organization Studies*.
4 Thousand Oaks, CA: Sage.
- 5 Lazer, D., Pentland, A. S., Adamic, L., Aral, S., Barabasi, A. L., et al. 2009. Life in the network:
6 the coming age of computational social science. *Science (New York, NY)*, 323(5915): 721.
- 7 Leenders, R. T. A., Contractor, N. S., & DeChurch, L. A. 2016. Once upon a time:
8 Understanding team processes as relational event networks. *Organizational Psychology*
9 *Review*, 6(1): 92–115.
- 10 Levitt, B., & March, J. G. 1988. Organizational learning. *Annual Review of Sociology*, 14(1):
11 319–338.
- 12 March, J. G. 1991. Exploration and exploitation in organizational learning. *Organization*
13 *Science*, 2(1): 71–87.
- 14 McCabe, T. J. 1976. A complexity measure. *IEEE Transactions on Software Engineering*, (4):
15 308–320.
- 16 McPherson, M., Smith-Lovin, L., & Cook, J. M. 2001. Birds of a Feather: Homophily in Social
17 Networks. *Annual Review of Sociology*, 27(1): 415–444.
- 18 Mesle, C., & Dibben, M. R. 2016. Whitehead's Process Relational Philosophy. *The SAGE*
19 *Handbook of Process Organization Studies*, 29.
- 20 Moe, N. B., Dingsøyr, T., & Dybå, T. 2010. A teamwork model for understanding an agile team:
21 A case study of a Scrum project. *Information and Software Technology*, 52(5): 480–491.
- 22 Moody, J., McFarland, D., & Bender-deMoll, S. 2005. Dynamic network visualization.
23 *American Journal of Sociology*, 110(4): 1206–1241.
- 24 Nicolini, D. 2009. Zooming in and out: Studying practices by switching theoretical lenses and
25 trailing connections. *Organization Studies*, 30(12): 1391–1418.
- 26 Obstfeld, D. 2012. Creative projects: A less routine approach toward getting new things done.
27 *Organization Science*, 23(6): 1571–1592.
- 28 Pentland, B. T. 1992. Organizing moves in software support hot lines. *Administrative Science*
29 *Quarterly*, 527–548.
- 30 Pentland, B. T. 1999. Building process theory with narrative: From description to explanation.
31 *Academy of Management Review*, 24(4): 711–724.
- 32 Pentland, B. T., & Feldman, M. S. 2007. Narrative networks: Patterns of technology and
33 organization. *Organization Science*, 18(5): 781–795.
- 34 Pentland, B. T., Feldman, M. S., Becker, M. C., & Liu, P. 2012. Dynamics of Organizational
35 Routines: A Generative Model. *Journal of Management Studies*, 49(8): 1484–1508.
- 36 Pentland, B. T., Hærem, T., & Hillison, D. 2010. Comparing organizational routines as recurrent
37 patterns of action. *Organization Studies*, 31(7): 917–940.
- 38 Pentland, B. T., & Liu, P. 2017. Network models of organizational routines: Tracing associations
39 between actions. In S. Jain & R. Mir (Eds.), *Routledge companion to qualitative research*
40 *in organization studies*.
- 41 Pentland, B. T., Liu, P., Kremser, W., & Hærem, T. In press. Dynamics of drift in digitized
42 processes. *MIS Quarterly*.
- 43 Pentland, B. T., Recker, J., & Wyner, G. 2015. *A thermometer for interdependence: Exploring*
44 *patterns of interdependence using networks of affordances*. Presented at the Thirty Sixth
45 International Conference on Information Systems, Fort Worth, TX, USA.
- 46 Pentland, B. T., Recker, J., & Wyner, G. 2016. *Conceptualizing and measuring*
47 *interdependence between organizational routines*. Presented at the Thirty Seventh
48 International Conference on Information Systems, Dublin, Ireland.
- 49
50
51
52
53
54
55
56
57
58
59
60

- 1
2
3 Pentland, B. T., Recker, J., & Wyner, G. 2017. Rediscovering Handoffs. *Academy of*
4 *Management Discoveries*, 3(3): 284–301.
- 5 Rerup, C., & Feldman, M. S. 2011. Routines as a source of change in organizational schemata:
6 The role of trial-and-error learning. *Academy of Management Journal*, 54(3): 577–610.
- 7 Rescher, N. 1996. *Process metaphysics: An introduction to process philosophy*. Albany: Suny
8 Press.
- 9 Rudolph, J. W., Morrison, J. B., & Carroll, J. S. 2009. The dynamics of action-oriented problem
10 solving: Linking interpretation and choice. *Academy of Management Review*, 34(4): 733–
11 756.
- 12 Salvato, C., & Rerup, C. 2018. Routine regulation: Balancing conflicting goals in organizational
13 routines. *Administrative Science Quarterly*, 63(1): 170–209.
- 14 Sarasvathy, S. D. 2001. Causation and effectuation: Toward a theoretical shift from economic
15 inevitability to entrepreneurial contingency. *Academy of Management Review*, 26(2): 243–
16 263.
- 17 Sarasvathy, S. D. 2009. *Effectuation: Elements of entrepreneurial expertise*. Edward Elgar
18 Publishing.
- 19 Schulz, M. 2008. Staying on track: a voyage to the internal mechanisms of routine reproduction.
20 *Handbook of organizational routines*: 228.
- 21 Shotter, J. 2006. Understanding process from within: An argument for ‘witness’-thinking.
22 *Organization Studies*, 27(4): 585–604.
- 23 Snijders, T. A. 2001. The statistical evaluation of social network dynamics. *Sociological*
24 *Methodology*, 31(1): 361–395.
- 25 Snijders, T. A. B., van de Bunt, G. G., & Steglich, C. E. G. 2010. Introduction to stochastic
26 actor-based models for network dynamics. *Social Networks*, 32(1): 44–60.
- 27 Song, M., Günther, C. W., & Van der Aalst, W. M. 2008. Trace clustering in process mining.
28 *International Conference on Business Process Management*, 109–120. Springer.
- 29 Srikanth, K., Harvey, S., & Peterson, R. 2016. A dynamic perspective on diverse teams: Moving
30 from the dual-process model to a dynamic coordination-based model of diverse team
31 performance. *The Academy of Management Annals*, 10(1): 453–493.
- 32 Strauss, A. L. 1993. *Continual permutations of action*. Aldine de Gruyter.
- 33 Strike, V. M., & Rerup, C. 2016. Mediated sensemaking. *Academy of Management Journal*,
34 59(3): 880–905.
- 35 Suchman, L. A. 1987. *Plans and situated actions: The problem of human-machine*
36 *communication*. Cambridge University Press.
- 37 Sutherland, J., & Sutherland, J. J. 2014. *Scrum: The Art of Doing Twice the Work in Half the*
38 *Time*. New York: Crown Business.
- 39 Sutton, R. S., & Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.
- 40 Tsoukas, H., & Chia, R. 2002. On organizational becoming: Rethinking organizational change.
41 *Organization Science*, 13(5): 567–582.
- 42 Turner, S. F., & Rindova, V. 2012. A balancing act: How organizations pursue consistency in
43 routine functioning in the face of ongoing change. *Organization Science*, 23(1): 24–46.
- 44 Turner, S. F., & Rindova, V. P. 2018. Watching the Clock: Action Timing, Patterning, and
45 Routine Performance. *Academy of Management Journal*, 61(4): 1253–1280.
- 46 Van de Ven, A. H., Polley, D., Garud, R., & Venkataraman, S. 1999. *The innovation journey*.
47 New York, NY: Oxford University Press.
- 48
49
50
51
52
53
54
55
56
57
58
59
60

- 1
2
3 Van de Ven, A. H., & Poole, M. S. 1995. Explaining development and change in organizations.
4 *Academy of Management Review*, 20(3): 510–540.
5 Volkoff, O., & Strong, D. M. 2017. Affordance theory and how to use it in IS research. *The*
6 *Routledge Companion to Management Information Systems*: 232–245. Routledge.
7 Wasserman, S., & Faust, K. 1994. *Social network analysis: Methods and applications*.
8 Cambridge University Press.
9 Weick, K. E. 1979. Cognitive processes in organizations. *Research in Organizational Behavior*,
10 1(1): 41–74.
11 West, D. B. 2001. *Introduction to graph theory*, vol. 2. Upper Saddle River: Prentice Hall.
12 Wood, R. E. 1986. Task complexity: Definition of the construct. *Organizational Behavior and*
13 *Human Decision Processes*, 37(1): 60–82.
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

Figure 1. Frequency of task types across Sprints

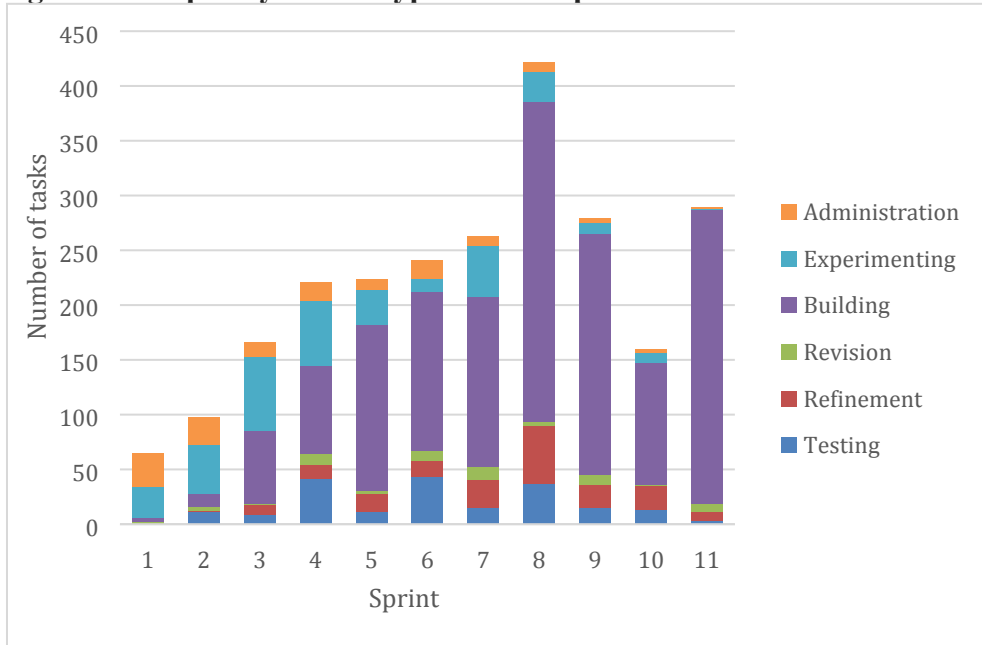
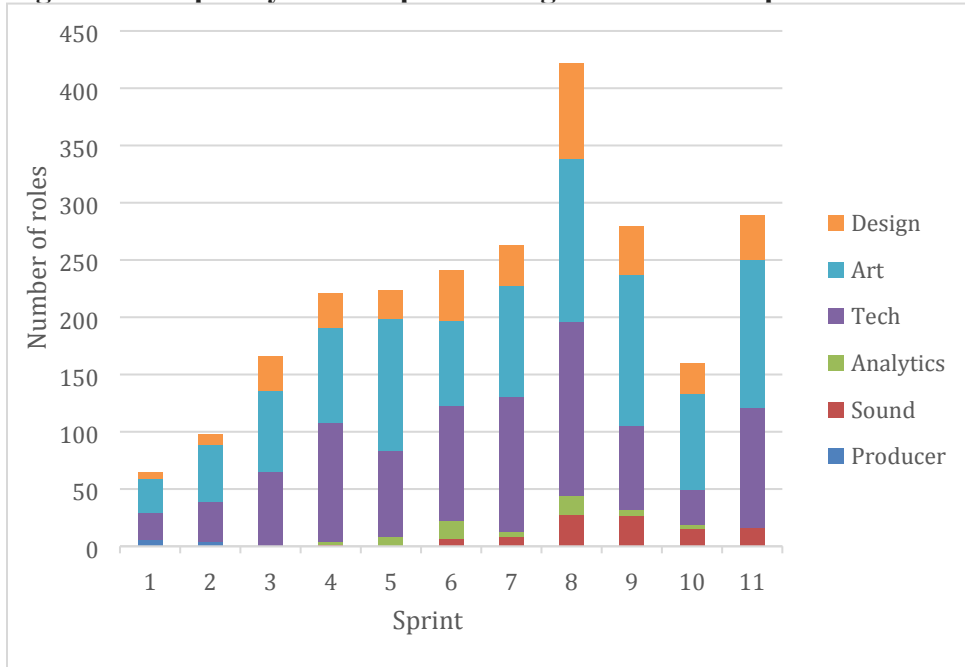


Figure 2. Frequency of roles performing a task across Sprints



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

Figure 3. Complexity index for sprints 1 to 11

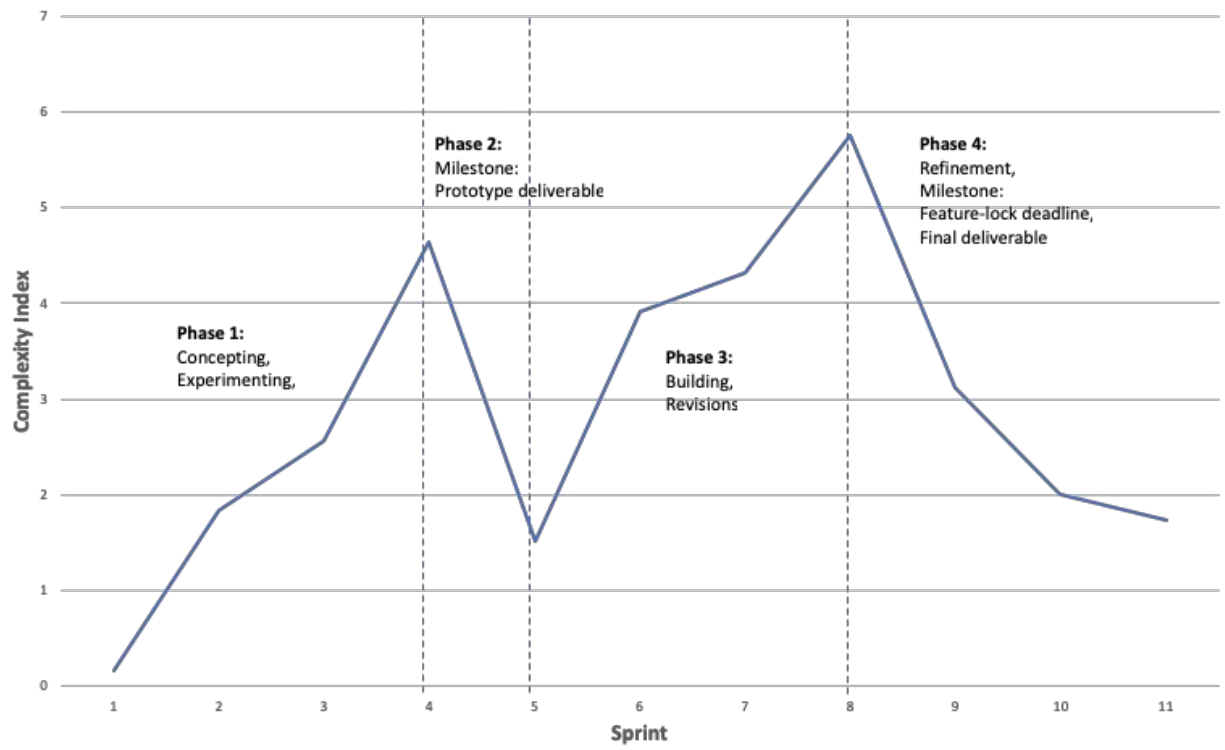


Figure 4a

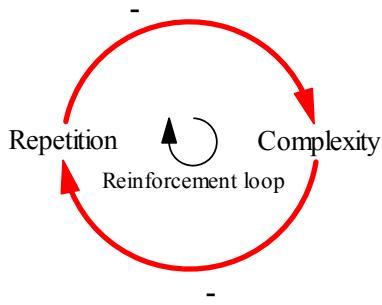


Figure 4d

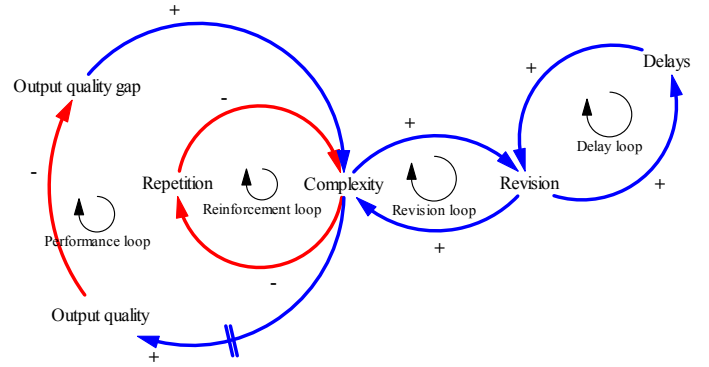


Figure 4b

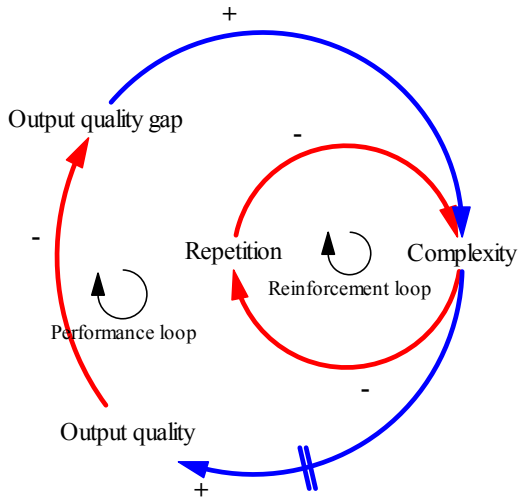


Figure 4e

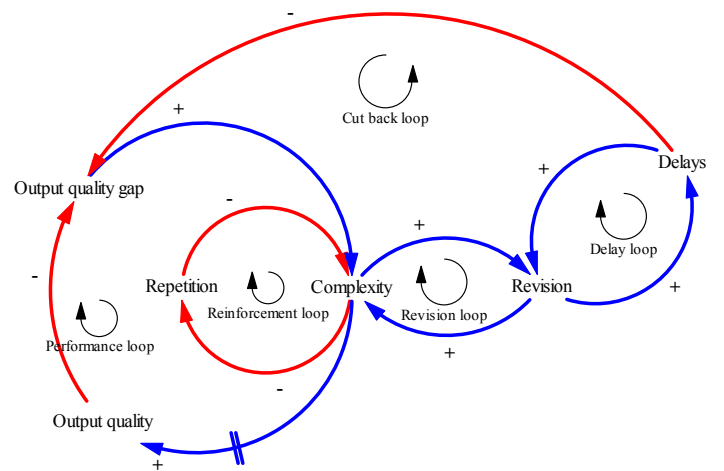


Figure 4c

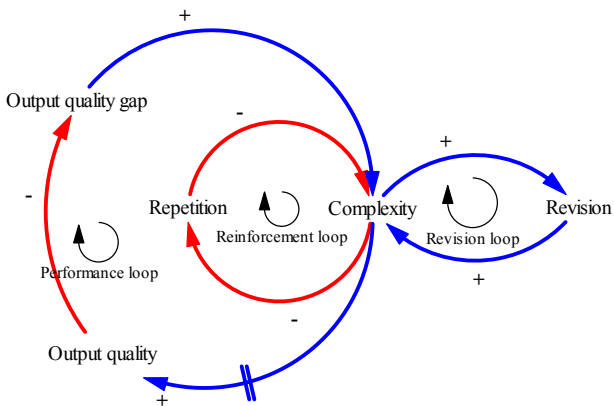
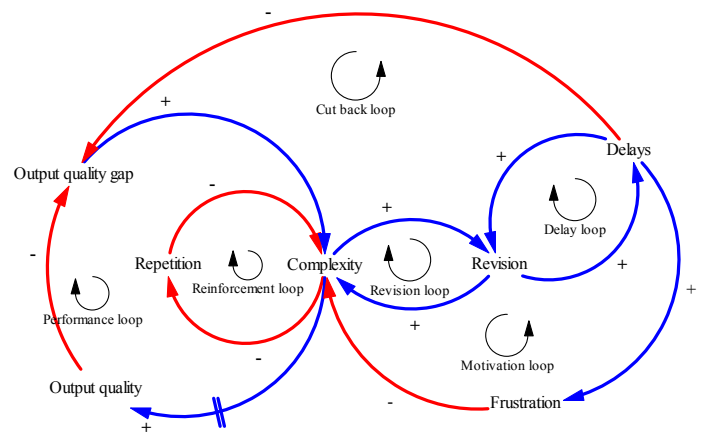


Figure 4f



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

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

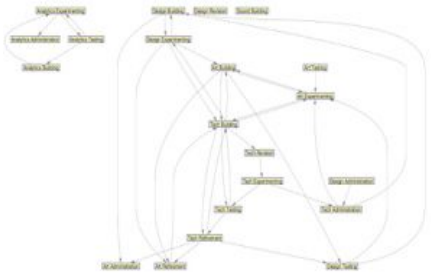
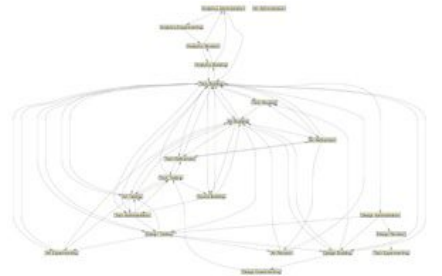
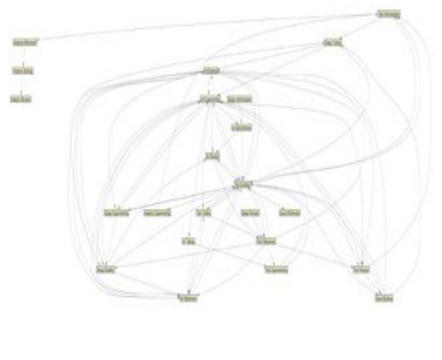
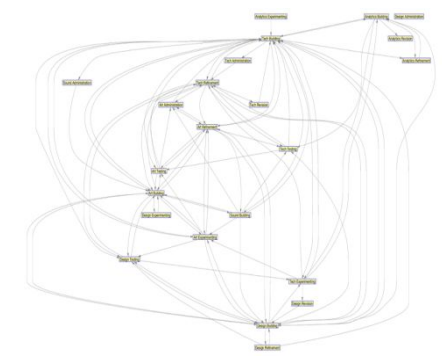
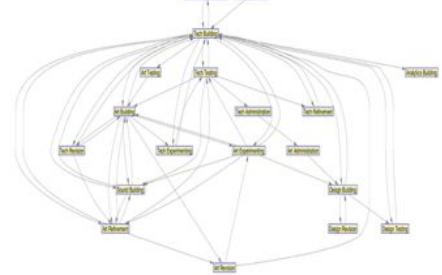
Table 2. Definition of task categories

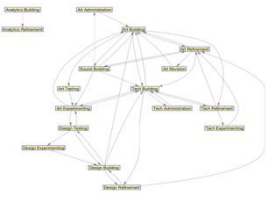
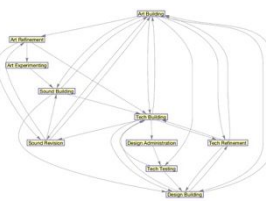
Category: Final	Category: Round 1	Definition
Administration	Administration	Activities that involve planning, organization, coordination, communication with internal or external parties.
Experimenting	Experimenting	Activities associated with learning, discovery, building experience or knowledge, addressing unanswered questions.
	Conceptualization	Activities associated with defining the form of team output. Includes definition of inter-relationships 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, reimplementation, 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 could 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/outcome is unintended.
Testing	Review	Reviewing work before release
	Testing	Activities directly associated with enacting playtests. Different from QA tests which checks 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.
	QA	Activities that involve testing for bugs, errors or edge cases. Different from playtests.

Table 3. Summary of mechanisms, complexity index, actions, handoffs, and graph diagrams.

Sprint	Graph	Complexity index	Actions (nodes)	Handoffs (edges)	Paths added/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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

<p>1 2 3 4 5 6 7 8 9 10 11 12</p> <p>5</p>		<p>1.52</p>	<p>17</p>	<p>47</p>	<p>-43618</p>	<p>Performance (reduction)</p>
<p>13 14 15 16 17 18 19 20 21 22</p> <p>6</p>		<p>3.92</p>	<p>21</p>	<p>83</p>	<p>8285</p>	<p>Performance Revision Delay Motivation</p>
<p>23 24 25 26 27 28 29 30 31 32 33 34</p> <p>7</p>		<p>4.32</p>	<p>21</p>	<p>86</p>	<p>12575</p>	<p>Performance Revision Delay Motivation</p>
<p>35 36 37 38 39 40 41 42 43 44 45 46</p> <p>8</p>		<p>5.76</p>	<p>21</p>	<p>106</p>	<p>554547</p>	<p>Performance Revision Delay Motivation</p>
<p>47 48 49 50 51 52 53 54 55 56 57 58 59 60</p> <p>9</p>		<p>3.12</p>	<p>18</p>	<p>66</p>	<p>-574122</p>	<p>Cut-back Performance</p>

10		2.0	16	51	-1218	Cut-back Performance
11		1.74	10	38	-45	Cut-back Performance

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
```

```

1
2
3
4 % OUTPUTS:
5 % simple_paths = number of simple paths
6 % list_of_unique_paths = cell array of strings that describe the paths
7
8
9 % Data structures for keep track of unique paths
10 paths_completed = containers.Map();
11 paths_in_progress = containers.Map();
12
13 % Use CAPITAL N for size of adjacency matrix
14 N = size(AM,1);
15
16 % Convert adjMtx to 0/1 only
17 AM = (AM > 0);
18
19 % Take out the diagonal, since self-connected vertices do not add paths
20 AM(eye(N)==1) = 0;
21
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 % The main loop traverses the graph until there are no more
24 % simple paths to find.
25 % Finished paths are stored in data.completed_paths
26
27 % start with the source to initialize the paths_in_progress
28 s = add_to_paths(source,next_nodes(AM, source));
29
30 % then loop until done
31 loop_count = 0;
32 simple_paths = 0;
33 while paths_in_progress.Count >= 1
34
35 % the function "path_search" loops through all of the paths in
36 % progress to see if they can be completed or continued.
37 % The status flag is not used
38 status_flag = path_search(AM);
39
40 % the loop count is just used to display progress if so desired
41 loop_count = loop_count+1;
42
43 % Uncomment these statements to view the process
44 % disp(strcat('Depth=', num2str(loop_count), ...
45 %   ' PathsInProgress:', num2str(paths_in_progress.Count),...
46 %   ' PathsCompleted:', num2str(paths_completed.Count)));
47 % show_paths(paths_in_progress)
48 % show_paths(paths_completed)
49
50 % If you want to set a ceiling, you can do it here.
51 if paths_completed.Count >= 1000000
52     % re-initialize the list of paths in progress to stop the search
53     paths_in_progress = containers.Map();
54     disp('*** over 1,000,000 paths found. Limiting count. ***');
55 end
56
57 end % paths_in_progress loop
58
59
60

```

```

1
2
3 % assign the total and the list
4 simple_paths = paths_completed.Count;
5 list_of_unique_paths = keys(paths_completed);
6
7 % disp(strcat('Total simple paths = ',num2str(simple_paths)));
8
9 return;
10
11 % *****
12
13 % loop through paths in progress and extend them until completed
14 function ss_status = path_search(adjMtx)
15
16     for p_in_prog = keys(paths_in_progress)
17         pnp = str2num(p_in_prog{1});
18         ss_status = add_to_paths(pnp, next_nodes(adjMtx,pnp));
19     end
20
21 end
22
23 % look at the end of the current path to get the next nodes
24 function nlist = next_nodes(adjMtx, current_path)
25
26     % use that node to get the list of next nodes
27     nlist = find(adjMtx(current_path(end),:) > 0);
28
29     % check if any are already on the path
30     nlist = setdiff(nlist,current_path);
31
32 end
33
34 % Put paths in paths_completed or paths_in_progress
35 % Keep going until all possible paths are found.
36 function sstatus = add_to_paths(path_in, next_node_list)
37
38     path_in_key = mat2str(path_in);
39
40     % stop if there is nowhere to go
41     if numel(next_node_list) == 0
42         if isKey(paths_in_progress, path_in_key)
43             remove(paths_in_progress, path_in_key);
44         end
45         sstatus = 0;
46         return;
47     else
48         sstatus = 1;
49     end
50
51     % loop through all of the potential next nodes
52     for next_node = next_node_list
53
54         % make sure the node has not been visited on this path
55         if ~any(path_in == next_node)
56
57             % append the next node and store it
58             path_out = [path_in, next_node];
59
60

```



```

1
2
3     path_out_key = mat2str(path_out);
4
5     % if the path is done, then save it in completed set
6     if path_out(end) == sink
7         paths_completed(path_out_key) = path_out;
8         sstatus = 1;
9     else
10        if numel(path_out) >= 3*N % path is too long...
11            if isKey(paths_in_progress, path_out_key)
12                remove(paths_in_progress, path_out_key);
13            end
14            sstatus = 0;
15        else
16            paths_in_progress(path_out_key) = 1; % dummy
17        end
18    end
19    if isKey(paths_in_progress, path_in_key)
20        remove(paths_in_progress, path_in_key);
21    end
22 end
23
24
25 % this function is only used for debugging
26 function show_paths(c)
27     for v = keys(c)
28         v
29     end
30 end
31
32
33 end % of main function
34

```

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 and Cain, 2016; Tiwari and Kumar, 2014). McCabe (1976) represents the execution paths in a block of code as a directed graph, and then uses the number of nodes and edges to estimate the number of execution pathways. McCabe also adjusts for subroutines, which act like nodes and tend to reduce the number of execution paths:

$$\textit{Complexity} \sim \textit{edges} - \textit{nodes} - \textit{subroutines}$$

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.

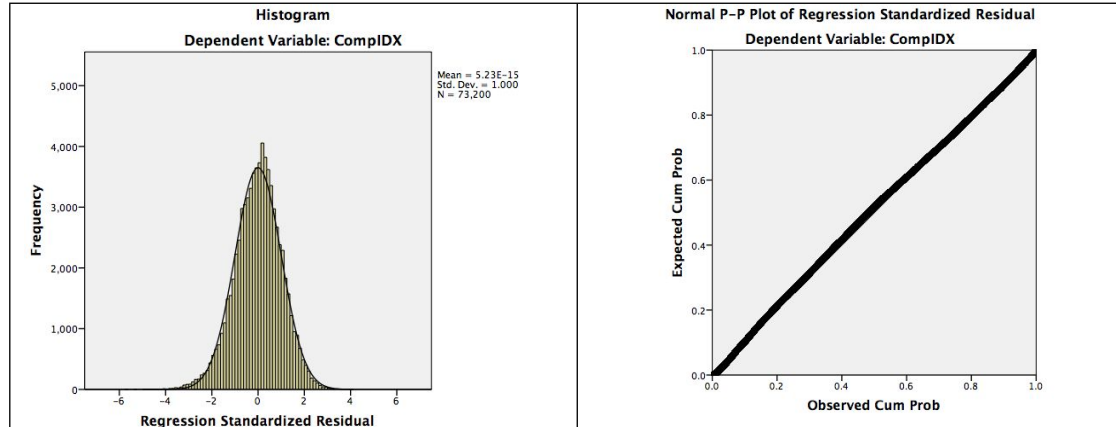
We fit this function to the results of the exact algorithm in Appendix A using a set of simulated data ($n=73,200$). The simulated data included 100 random repetitions for each size of network from $10 \leq \text{nodes} \leq 100$, with varying levels of network density. The networks were simulated so 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 count the number of paths or the sum of the number of edges along all of the paths. The results are shown in Table B-1. Regression diagnostics are shown in Figure B-2. Over a wide range of conditions, the simple estimate of network paths correlates with the exact count quite well ($r=0.94$).

Table B-1: Fitting the estimate to the exact model

Descriptive Statistics (N=73200)			
	Min	Avg	Max
Nodes	10	54	100
Edges	9	67	138
Paths	1	120	25838

	DV = $\text{Log}_{10}(\text{simple paths})$
Nodes	-0.079*** (.000)
Edges	0.080*** (.000)
Const	0.120** (.003)
R ²	0.889
F	291982.5***

Figure B-1: Diagnostic results



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, $\log_{10}(1) = 0$. 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:

$$\text{Enacted task complexity} = \log_{10}(\text{simple paths}) = .08 * (\text{edges} - \text{nodes} + 1)$$

Bax, E. T. 1994. Algorithms to count paths and cycles. *Information Processing Letters*, 52(5): 249-252.

Roberts, B., & Kroese, D. P. 2007. Estimating the Number of st Paths in a Graph. *Journal of Graph Algorithms and Applications*, 11(1): 195-214.

Rubin, F. 1978. Enumerating all simple paths in a graph. *IEEE Transactions on Circuits and Systems*, 25(8): 641-642.

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*, *ReverbNation*, *YouTube* and elsewhere.